

[illegible][illegible]

```
UU      UU  EEEEEEEEE TTTTTTTTT DDDDDDDD RRRRRRRR 77777777 888888 000000 000000
UU      UU  EEEEEEEEE TTTTTTTTT DDDDDDDD RRRRRRRR 77777777 888888 000000 000000
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EEEEEEEEE TT          DD      DD  RRRRRRRR 77      88      00      00
UU      UU  EEEEEEEEE TT          DD      DD  RRRRRRRR 77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UU      UU  EE          TT          DD      DD  RR      RR  77      88      00      00
UUUUUUUUU EEEEEEEEE TT          DDDDDDDD RR      RR  77      888888 000000 000000
UUUUUUUUU EEEEEEEEE TT          DDDDDDDD RR      RR  77      888888 000000 000000
                                     ....
                                     ....
                                     ....
                                     ....

LL      IIIIII SSSSSSSS
LL      IIIIII SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLL IIIIII SSSSSSSS
```



(2)	88	Declarations
(3)	223	Read-Only Data
(4)	427	Read/Write Data
(4)	599	COMMAND BLOCK & PACKETS
(5)	927	RMS-32 Data Structures
(6)	990	Main Program
(11)	1432	Test the DR780/DR750
(14)	1622	RANBUF
(14)	1647	BADQUEUE
(14)	1674	PKT_CHECK
(15)	1741	System Service Exception Handler
(16)	1870	RMS Error Handler
(17)	1934	CTRL/C Handler
(18)	1979	Error Exit
(19)	2041	Exit Handler

```

0000 1      .TITLE UETDR7800 VAX/VMS UETP DEVICE TEST FOR DR780/DR750
0000 2      .IDENT 'V04-000'
0000 3      .ENABLE SUPPRESSION
0000 4      :
0000 5      :*****
0000 6      :*
0000 7      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      :*  ALL RIGHTS RESERVED.
0000 10     :*
0000 11     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     :*  TRANSFERRED.
0000 17     :*
0000 18     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     :*  CORPORATION.
0000 21     :*
0000 22     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     :*
0000 25     :*
0000 26     :*****
0000 27     :
0000 28     :
0000 29     :++
0000 30     : FACILITY:
0000 31     :   This module will be distributed with VAX/VMS under the [SYSTEST]
0000 32     :   account.
0000 33     :
0000 34     : ABSTRACT:
0000 35     :   This module exercises a DR780 or a DR750 in the VAX/VMS system using
0000 36     :   QIO functions. 2048(10) byte transfers are written and read
0000 37     :   using chained command packets. The transfers are verified for
0000 38     :   correct data.
0000 39     :
0000 40     : ENVIRONMENT:
0000 41     :   This program will run in user access mode, with interrupts enabled
0000 42     :   at all times. This program must be linked with SYS$SYSTEM:SYS.STB
0000 43     :   because of its use of symbol IOC$GW_XFMXRATE to get the current DR
0000 44     :   max transfer rate. This program requires the following privileges and
0000 45     :   quotas:
0000 46     :
0000 47     : --
0000 48     :
0000 49     : AUTHOR: Larry Jones,  CREATION DATE: May, 1981
0000 50     :
0000 51     : MODIFIED BY:
0000 52     :
0000 53     :   V03-010 RNH0008      Richard N. Holstein,    21-Mar-1984
0000 54     :   Change wording on error messages.
0000 55     :
0000 56     :   V03-009 RNH0007      Richard N. Holstein,    15-Feb-1984
0000 57     :   Take advantage of the new UETP message codes.  Fix SSERROR

```



0000	58	:	interaction with RMS_ERROR.
0000	59	:	
0000	60	:	V03-008 RNH0006 Richard N. Holstein, 05-Jan-1984
0000	61	:	Set up SYS\$ERROR for the ucode loader process and report back
0000	62	:	any results.
0000	63	:	
0000	64	:	V03-007 RNH0005 Richard N. Holstein, 19-Dec-1983
0000	65	:	Give correct sentinels to Test Controller.
0000	66	:	
0000	67	:	V03-006 RNH0004 Richard N. Holstein, 21-Nov-1983
0000	68	:	Use decimal conversion routine for unit numbers.
0000	69	:	
0000	70	:	V03-005 RNH0003 Richard N. Holstein, 11-Mar-1983
0000	71	:	Don't signal ending message in EXIT_HANDLER.
0000	72	:	
0000	73	:	V03-004 RNH0002 Richard N. Holstein, 25-Feb-1983
0000	74	:	Allow for longer device names.
0000	75	:	
0000	76	:	V03-003 RNH0001 Richard N. Holstein, 15-Oct-1982
0000	77	:	Miscellaneous fixes listed in the V3B UETP Workplan.
0000	78	:	
0000	79	:	V03-002 LDJ0002 Larry D. Jones, 11-Mar-1982
0000	80	:	Fixed missing bit set in command table for DR.
0000	81	:	
0000	82	:	V03-001 LDJ0001 Larry D. Jones, 29-Sep-1981
0000	83	:	Filled in error path exits with missing STATUS values and
0000	84	:	reversed the order of the error/end test.
0000	85	:	
0000	86	: **	

```
0000 88      .SBTTL Declarations
0000 89      :
0000 90      : INCLUDE FILES:
0000 91      :
0000 92      : SYSS$LIBRARY:LIB.MLB      for general definitions
0000 93      : SHRLIB$:UETP.MLB        for UETP definitions
0000 94      :
0000 95      :
0000 96      : MACROS:
0000 97      :
0000 98      : $ACCTDEF                  ; Accounting definitions
0000 99      : $SCHFDEF                 ; Condition handler frame definitions
0000 100     : $DEVDEF                  ; Device definitions
0000 101     : $DIBDEF                  ; Device Information Block
0000 102     : $SDVIDEF                 ; $GETDVI ITMLST item codes
0000 103     : $SHRDEF                  ; Shared messages
0000 104     : $SSDEF                   ; System Service status codes
0000 105     : $STSDEF                  ; Status return
0000 106     : $UETUNTDEF               ; UETP unit block offset definitions
0000 107     : $UETPDEF                 ; UETP
0000 108     : $XFDEF                   ; DR780 definitions
0000 109     :
0000 110     : USER MACRO DEFINITIONS
0000 111     :
0000 112     : QRETRY - This macro executes an interlocked queue instruction and
0000 113     :               retries up to 25 times if the queue is locked.
0000 114     : INPUTS:
0000 115     :   OPCODE = OPCODE NAME : INSQHI, INSQTI, REMQHI, REMQTI.
0000 116     :   OPERAND1 = first operand for opcode.
0000 117     :   OPERAND2 = second operand for opcode.
0000 118     :   SUCCESS = label to branch to if operation succeeds (may be defaulted).
0000 119     :   ERROR   = label to branch to if operation fails (may be omitted).
0000 120     :
0000 121     : OUTPUTS:
0000 122     :   R0      = destroyed.
0000 123     :   C-BIT  = clear if operation succeeded.
0000 124     :               set if operation failed - queue locked.
0000 125     :               (must be checked before V-bit or Z-bit)
0000 126     :
0000 127     : REMQTI OR REMQHI:
0000 128     :
0000 129     : V-bit = clear if an entry removed from queue.
0000 130     :               set if no entry removed from the queue.
0000 131     :
0000 132     : INSQTI OR INSQHI:
0000 133     :
0000 134     : Z-bit = clear if entry is not first in the queue.
0000 135     :               set if entry is first in the queue.
0000 136     :
0000 137     : .MACRO QRETRY OPCODE,OPERAND1,OPERAND2,SUCCESS,ERROR,?LOOP,?OK
0000 138     : CLRL      R0
0000 139     : LOOP:
0000 140     :   OPCODE  OPERAND1,OPERAND2
0000 141     :   .IF      NB      SUCCESS
0000 142     :   BCC      SUCCESS
0000 143     :   .IFF
0000 144     :   BCC      OK
```



```
0000 145 .ENDC
0000 146 AOBLS #25,R0,LOOP
0000 147 .IF NB ERROR
0000 148 BRW ERROR
0000 149 .ENDC
0000 150 OK:
0000 151 .ENDM QRETRY
0000 152 ;
0000 153 .MACRO BUILD,NAME,STATUS
0000 154 .PC1... ; PC of ASCII pkt NAME table
0000 155 .ADDRESS PC2... ; ASCII pkt address
0000 156 .LONG ^X'STATUS ; Expected packet return code
0000 157 PC1...=PC1...+8 ; Bump to the next address
0000 158 .PC2... ; Point to the next ASCII msg
0000 159 ; Make it's label and function ID
0000 160 NAME:
0000 161 .ASCII /NAME, /
0000 162 PC2...= ; Update the string PC
0000 163 .ENDM BUILD
0000 164 ;
0000 165 ; EQUATED SYMBOLS:
0000 166 ;
0000 167 ; Facility number definitions:
00000001 0000 168 RMS$_FACILITY = 1
0000 169
0000 170 ; SHR message definitions:
00740000 0000 171 UETP = UETPS$_FACILITY@STSSV_FAC_NO ; Define the UETP facility code
007410E0 0000 172 UETPS$_ABENDD = UETP!SHRS$_ABENDD ; Define the UETP message codes
00741038 0000 173 UETPS$_BEGIN = UETP!SHRS$_BEGIN
00741080 0000 174 UETPS$_ENDD = UETP!SHRS$_ENDD
00741098 0000 175 UETPS$_OPENIN = UETP!SHRS$_OPENIN
00741130 0000 176 UETPS$_TEXT = UETP!SHRS$_TEXT
0000 177
0000 178 ; Internal flag bits...:
00000000 0000 179 CC_FLGV = 0 ; Set when a control C is typed
00000001 0000 180 TEST_OVERV = 1 ; Set when test is over
00000002 0000 181 SAFE_TO_UPDV = 2 ; Set if it's safe to update UETINIDEV
00000003 0000 182 ERR_FLGV = 3 ; Set when an error occurs
00000004 0000 183 FPAC_FLGV = 4 ; Set when first packet is serviced
00000005 0000 184 BEGIN_MSGV = 5 ; Set if 'BEGIN' msg has been printed
0000 185
0000 186 ; ...and corresponding masks:
00000001 0000 187 CC_FLGM = 1@CC_FLGV
00000002 0000 188 TEST_OVERM = 1@TEST_OVERV
00000004 0000 189 SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008 0000 190 ERR_FLGM = 1@ERR_FLGV
00000010 0000 191 FPAC_FLGM = 1@FPAC_FLGV
00000020 0000 192 BEGIN_MSGM = 1@BEGIN_MSGV
0000 193
0000 194 ; Miscellany:
00000020 0000 195 LC_BITM = ^X20 ; Mask to convert lower case to upper
0000001B 0000 196 ESC = ^X1B ; Escape definition
00000028 0000 197 REC_SIZE = 40 ; UETINIDEV.DAT record size
00000084 0000 198 TEXT_BUFFER = 132 ; Internal text buffer size
00000003 0000 199 SS_SYNCH_EFN = 3 ; Synch miscellaneous system services
0000000F 0000 200 MAX_PROC_NAME = 15 ; Longest process name
0000000A 0000 201 MAX_DEV_DESIG = 10 ; Longest possible controller name
```

00000005	0000	202	MAX_UNIT_DESIG= 5	; Longest possible unit number
00000011	0000	203	NO_OF_POS_PKTS= 17	; Number of possible packet types
0000000D	0000	204	PKT_COUNT = 13	; Number of packets to be processed
00006030	0000	205	UNUSED_FUNC = ^X6030	; Bit mask of the unused functions
00000800	0000	206	BUFSIZ = 2048	; Read/write buffer size
00000001	0000	207	EFN1 = 1	; EF number definitions
00000004	0000	208	EFN2 = 4	; EFN used for three minute timer
	0000	209		
	0000	210	; The following definitions are set depending on the device under test.	
	0000	211		
00000000	0000	212	DEVDEP_SIZE = 0	; Size of device dependent part of UETUNT
00000800	0000	213	WRITE_SIZE = BUFSIZ	; Size of device write buffer
00000800	0000	214	READ_SIZE = BUFSIZ	; Size of device read buffer
	0000	215		
	0000	216	PAGES = <<UETUNT\$C INDSIZ+-	; Add together all of the pieces...
	0000	217	DEVDEP_SIZE+-	; ...which make up a UETP unit block...
	0000	218	WRITE_SIZE+-	; ...to give to the \$EXPREG service below
	0000	219	READ_SIZE+-	
00000009	0000	220	511>7512>	
	0000	221		



```
0000 223 .SBTTL Read-Only Data
0000 224 .PSECT RODATA,NOEXE,NOWRT,PAGE
0000 225
53 45 54 53 59 53 00000008'010E0000' 0000 226 ACNT_NAME: ; Process name on exit
54 000E 227 .ASCID /SYSTEST/
000F 228
000F 229 TEST_NAME: ; This test name
37 52 44 54 45 55 00000017'010E0000' 000F 230 .ASCID /UETDR7800/
30 30 38 001D 231
0020 232 SUPDEV_GBLSEC: ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000' 0020 233 .ASCID /UETSUPDEV/
56 45 44 002E 234
0031 235 CONTROLLER: ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000' 0031 236 .ASCID /CTRLNAME/
45 4D 003F 237
0041 238 PROCESS: ; ucode load program
59 53 24 53 59 53 00000049'010E0000' 0041 239 .ASCID /SYS$SYSTEM:XFLOADER.EXE/
45 44 41 4F 4C 46 58 3A 4D 45 54 53 004F 240
45 58 45 2E 52 005B 241 XFLDR_SYSS$ERROR: ; File name of SYSS$ERROR for XFLOADER
0060 242 .ASCII /XFLDR_ERROR.LOG/
0060 243 XFLDR_SYSS$ERROR_LENGTH = .-XFLDR_SYSS$ERROR
006C 244
006F 245 XFLDR_SYSS$ERROR_DESC: ; SYSS$ERROR descriptor during $CREPRC
0000 000F 006F 246 .WORD XFLDR_SYSS$ERROR_LENGTH,0
00000060' 0073 247 .ADDRESS XFLDR_SYSS$ERROR
0077 248
0077 249 XFLDR_HUNG: ; We timed out waiting to load ucode
6D 20 32 33 52 44 0000007F'010E0000' 0077 250 .ASCID /DR32 microcode loader process seems to be hung. /
61 6F 6C 20 65 64 6F 63 6F 72 63 69 0085
20 73 73 65 63 6F 72 70 20 72 65 64 0091
20 65 62 20 6F 74 20 73 6D 65 65 73 009D
20 2E 67 6E 75 68 00A9
00AF 251
00AF 252 XFLDR_LOG: ; Error messages during ucode loading
6D 20 32 33 52 44 000000B7'010E0000' 00AF 253 .ASCID /DR32 microcode loader process logged some error message(s): /
61 6F 6C 20 65 64 6F 63 6F 72 63 69 00BD
20 73 73 65 63 6F 72 70 20 72 65 64 00C9
20 65 6D 6F 73 20 64 65 67 67 6F 6C 00D5
67 61 73 73 65 6D 20 72 6F 72 72 65 00E1
20 3A 29 73 28 65 00ED
00F3 254
00F3 255 XFLDR_COPY_START: ; $PUTMSG MSGVEC for start copying log
000F 0006 00F3 256 .WORD 6,^XF
00741132 00F7 257 .LONG UETPS_TEXT!ST$K_ERROR
0000 0001 00FB 258 .WORD 1,0
000000AF' 00FF 259 .ADDRESS XFLDR_LOG
007480B1 0103 260 .LONG UETPS_COPY_LOG
0000 0001 0107 261 .WORD 1,0
0000006F' 010B 262 .ADDRESS XFLDR_SYSS$ERROR_DESC
010F 263
```



```
0001 0004 010F 264 XFLDR_COPY LINE: ; $PUTMSG MSGVEC for copying log line
007480B9 010F 265 .WORD 4,^X1
0000 0002 0113 266 .LONG UETPS_COPY_LOG_LINE
00000004 0117 267 .WORD 2,0
00000014 011B 268 .LONG 4
011F 269 .ADDRESS BUFFER_PTR
0123 270
0123 271 XFLDR_COPY FINISH: ; $PUTMSG MSGVEC for ucode !r log end
000F 0003 0123 272 .WORD 3,^XF
007480C1 0127 273 .LONG UETPS_COPY_LOG_ENDED
0000 0001 012B 274 .WORD 1,0
0000006F 012F 275 .ADDRESS XFLDR_SYSS$ERROR_DESC
0133 276
0133 277 MODE: ; Run mode logical name
45 44 4F 4D 0000013B'010E0000' 0133 278 .ASCID /MODE/
013F 279
013F 280 CS:
52 44 20 64 61 42 00000147'010E0000' 013F 281 .ASCID /Bad DR !AC packet DSL of !XL expected !XL/
20 74 65 6B 63 61 70 20 43 41 21 20 014D
65 20 4C 58 21 20 66 6F 20 4C 53 44 0159
4C 58 21 20 64 65 74 63 65 70 78 0165
0170 282
0170 283 NO_RMS_AST TABLE: ; List of errors for which...
00000000' 0170 284 .LONG RMS$_BLN ; ...RMS cannot deliver an AST...
00000000' 0174 285 .LONG RMS$_BUSY ; ...even if one has an ERR= arg
00000000' 0178 286 .LONG RMS$_CDA ; Note that we can search table...
00000000' 017C 287 .LONG RMS$_FAB ; ...via MATCHC since <31:16>...
00000000' 0180 288 .LONG RMS$_RAB ; ...pattern can't be in <15:0>
00000014 0184 289 NRAT_LENGTH = .-NO_RMS_AST_TABLE
0184 290
0184 291 SYSS$INPUT: ; Name of device from which...
4E 49 24 53 59 53 0000018C'010E0000' 0184 292 .ASCID /SYSS$INPUT/
54 55 50 0192
0195 293
0195 294 INPUT_ITMLST: ; $GETDVI arg list for SYSS$INPUT
0020 0040 0195 295 .WORD 64,DVIS$DEVNAM ; We need the equivalence name
00000014'0000001C' 0199 296 .LONG BUFFER,BUFFER_PTR
00000000 01A1 297 .LONG 0 ; Terminate the list
01A5 298
01A5 299 CS1: ; Device class and type control string
21 20 42 58 32 21 000001AD'010E0000' 01A5 300 .ASCID /!2XB !2XB /
20 42 58 32 01B3
01B7 301
01B7 302 CS2:
52 44 20 65 68 54 000001BF'010E0000' 01B7 303 .ASCID /The DR!AC data rate is !XW which is !AS megabytes per second./
74 61 72 20 61 74 61 64 20 43 41 21 01C5
69 68 77 20 57 58 21 20 73 69 20 65 01D1
65 6D 20 53 41 21 20 73 69 20 68 63 01DD
20 72 65 70 20 73 65 74 79 62 61 67 01E9
2E 64 6E 6F 63 65 73 01F5
01FC 304
01FC 305 CS3: ; Device class-only control string
2A 20 42 58 32 21 00000204'010E0000' 01FC 306 .ASCID /!2XB **/
2A 020A
020B 307
020B 308 CS4:
000000C4' 020B 309 .LONG CS1L
```



```
00000213' 020F 310 .ADDRESS +4
63 20 74 65 6B 63 61 70 20 64 61 42 0213 311 .ASCII \Bad packet count, expected !UL packets but received !UL!/\
74 63 65 70 78 65 20 2C 74 6E 75 6F 021F
65 6B 63 61 70 20 4C 55 21 20 64 65 022B
69 65 63 65 72 20 74 75 62 20 73 74 0237
2F 21 2E 4C 55 21 20 64 65 76 0243
20 67 6E 69 73 73 69 4D 5F 21 5F 21 024D 312 .ASCII \!_!_Missing !UL!- packet!%S : !#(AC)\
74 65 6B 63 61 70 20 2D 21 4C 55 21 0259
29 43 41 28 23 21 20 3A 20 53 25 21 0265
000000C4 0271 313 CS1L=-CS1-8
0271 314
0271 315 ULOAD_FAILED:
6F 63 75 20 52 44 00000279'010E0000' 0271 316 .ASCII \DR ucode load failure.\
6C 69 61 66 20 64 61 6F 6C 20 65 64 027F
2E 65 72 75 028B
028F 317
028F 318 START_DATA_FAILED:
64 65 6C 69 61 46 00000297'010E0000' 028F 319 .ASCII \Failed to start data transfer\
61 64 20 74 72 61 74 73 20 6F 74 20 029D
72 65 66 73 6E 61 72 74 20 61 74 02A9
02B4 320
02B4 321 CNTRLMSG:
65 74 72 6F 62 41 000002BC'010E0000' 02B4 322 .ASCII \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64 02C2
43 2F 4C 52 54 43 20 02CE
02D5 323
02D5 324 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000002DD'010E0000' 02D5 325 .ASCII /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74 02E3
2E 64 65 69 66 69 02EF
02F5 326
02F5 327 DEAD_CTRLNAME:
20 74 27 6E 61 43 000002FD'010E0000' 02F5 328 .ASCII /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74 0303
72 61 6D 20 2C 53 41 21 20 72 65 6C 030F
61 73 75 6E 75 20 73 61 20 64 65 6B 031B
4E 49 54 45 55 20 6E 69 20 65 6C 62 0327
2E 54 41 44 2E 56 45 44 49 0333
033C 329
033C 330 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000344'010E0000' 033C 331 .ASCII /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74 034A
2E 67 6E 69 74 73 65 74 20 72 6F 66 0356
0362 332
0362 333 ILLEGAL_REC:
61 67 65 6C 6C 49 0000036A'010E0000' 0362 334 .ASCII /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C 0370
20 65 6C 69 66 20 6E 69 20 74 61 6D 037C
41 44 2E 56 45 44 49 4E 49 54 45 55 0388
21 54 0394
0396 335
0396 336 PASS_MSG:
66 6F 20 64 6E 45 0000039E'010E0000' 0396 337 .ASCII /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20 03A4
61 72 65 74 69 20 4C 55 21 20 68 74 03B0
44 25 21 20 74 61 20 73 6E 6F 69 74 03BC
2E 03C8
```



```
03C9 338
03C9 339 DR780: ; DR780 name
30 38 37 00' 03C9 340 .ASCIC /780/
03 03C9
03CD 341 DR750: ; DR750 name
30 35 37 00' 03CD 342 .ASCIC /750/
03 03CD
03D1 343
03D1 344 INIDEV_UPDERR: ; Error during exit handler
03D1 345 .ASCID /Error updating UETINIDEV.DAT./
20 72 6F 72 72 45 000003D9'010E0000' 03DF
54 45 55 20 67 6E 69 74 61 64 70 75 03EB
2E 54 41 44 2E 56 45 44 49 4E 49 03F6
03F6 346
03F6 347 THREEMIN: ; 3 minute delta time
FFFFFFF 94B62E00 03F6 348 .LONG -10*1000*1000*180,-1
03FE 349 ONEMIN: ; 1 minute delta time
FFFFFFF DC3CBA00 03FE 350 .LONG -10*1000*1000*60,-1
0406 351 TENSEC: ; 10 second delta time
FFFFFFF FA0A1F00 0406 352 .LONG -10*1000*1000*10,-1
040E 353
040E 354 UNIT_DESC: ; Descriptor used to convert unit #
00000005 040E 355 .LONG 5
00000022' 0412 356 .ADDRESS BUFFER+6
0416 357
0416 358 CONT_DESC: ; Descriptor used to convert controller...
0000 0028 0416 359 .WORD REC_SIZE,0
0000001C' 041A 360 .ADDRESS BUFFER
041E 361
041E 362 FILE: ; Fills in RMS_ERR_STRING
65 6C 69 66 00000426'010E0000' 041E 363 .ASCID /file/
042A 364
042A 365 RECORD: ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000432'010E0000' 042A 366 .ASCID /record/
0438 367
0438 368 RMS_ERR_STRING: ; Announces an RMS error
41 21 20 53 4D 52 00000440'010E0000' 0438 369 .ASCID /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53 0446
44 41 21 20 65 6C 69 0452
0459 370
0459 371 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43 0459 372 .ASCII /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65 0465
20 0471
00000019 0472 373 PMTSIZ = .-PROMPT
0472 374
0472 375 BADQUE:
75 71 20 64 61 42 0000047A'010E0000' 0472 376 .ASCID /Bad queue entry detected! Fatal error!/
65 64 20 79 72 74 6E 65 20 65 75 65 0480
61 74 61 46 20 21 64 65 74 63 65 74 048C
21 72 6F 72 72 65 20 6C 0498
04A0 377
04A0 378 TEST_HUNG:
6E 75 68 20 52 44 000004A8'010E0000' 04A0 379 .ASCID /DR hung, check backplane jumpers needed for testing./
63 61 62 20 6B 63 65 68 63 20 2C 67 04AE
65 70 6D 75 6A 20 65 6E 61 6C 70 6B 04BA
6F 66 20 64 65 64 65 65 6E 20 73 72 04C6
2E 67 6E 69 74 73 65 74 20 72 04D2
```



```
000012D8' 04DC 380
000015BC' 04DC 381 CMDBLKDES:
04E0 382 .ADDRESS CMDBLK ; LKWSET parameter list for locking
04E4 383 .ADDRESS CMDBLKEND ; Down the command block
04E4 384
04E4 385 NAME_TBL: ; Table of pointers to ASCII packet names
04E4 386 ; And expected packet status returns
0000056C 04E4 387 .BLKL NO_OF_POS_PKTS*2 ; ** NOTE ** table must be in this order
056C 388
056C 389 PKT_TBL: ; Table of pointers to DR packets
00001598' 056C 390 .ADDRESS FREE_PKT ; ** NOTE ** table must be in this order
000012F0' 0570 391 .ADDRESS NOOP_PKT
00001330' 0574 392 .ADDRESS SET_SELF_PKT
00001370' 0578 393 .ADDRESS DIAG_WRI_PKT
000013B0' 057C 394 .ADDRESS READ_DDI_PKT
00001390' 0580 395 .ADDRESS DIAG_REA_PKT
000013D0' 0584 396 .ADDRESS WRITE_CH_PKT
00001400' 0588 397 .ADDRESS WRITE_PKT
000014A0' 058C 398 .ADDRESS READ_CHA_PKT
000014D0' 0590 399 .ADDRESS READ_PKT
00001570' 0594 400 .ADDRESS DIAG_WRT_PKT
00001350' 0598 401 .ADDRESS CLR_SELF_PKT
00001310' 059C 402 .ADDRESS HALT_PKT
05A0 403
000004E4 05A0 404 PC1...=NAME_TBL ; Set name pointer
000005A0 05A0 405 PC2...= ; Set name table pointer
05A0 406
05A0 407 .LIST MEB
05A0 408 BUILD READ,00000023 ; Build the tables and the type name
000004E4 05A0 ; PC of ASCII pkt READ table
000005A0' 04E4 .ADDRESS PC2... ; ASCII pkt address
00000023 04E8 .LONG ^X00000023 ; Expected packet return code
000005A0 04EC .=PC2... ; Point to the next ASCII msg
05A0
20 2C 44 41 45 52 00' 05A0 READ: .ASCII /READ, /
06 05A0
05A7 409 BUILD READ_CHAIN,00000023 ; PC of ASCII pkt READ_CHAIN table
000004EC 05A7 .=PC1... ; ASCII pkt address
000005A7' 04EC .ADDRESS PC2... ; Expected packet return code
00000023 04F0 .LONG ^X00000023 ; Point to the next ASCII msg
000005A7 04F4 .=PC2...
05A7 READ_CHAIN: .ASCII /READ_CHAIN, /
2C 4E 49 41 48 43 5F 44 41 45 52 00' 05A7
20 05B3
0C 05A7
05B4 410 BUILD WRITE,00000023 ; PC of ASCII pkt WRITE table
000004F4 05B4 .=PC1... ; ASCII pkt address
000005B4' 04F4 .ADDRESS PC2... ; Expected packet return code
00000023 04F8 .LONG ^X00000023 ; Point to the next ASCII msg
000005B4 04FC .=PC2...
05B4 WRITE: .ASCII /WRITE, /
20 2C 45 54 49 52 57 00' 05B4
07 05B4
05BC 411 BUILD WRITE_CHAIN,00000023 ; PC of ASCII pkt WRITE_CHAIN table
000004FC 05BC .=PC1... ; ASCII pkt address
000005BC' 04FC .ADDRESS PC2... ; Expected packet return code
00000023 0500 .LONG ^X00000023
```



```
000005BC 0504
4E 49 41 48 43 5F 45 54 49 52 57 00' 05BC
20 2C 05BC
0D 05C8
05BC
05CA 412 BUILD WRITE_DEV_CNTRL,00000023
00000504 05CA ; PC of ASCII pkt WRITE_DEV_CNTRL table
000005CA' 0504 ; ASCII pkt address
00000023 0508 ; Expected packet return code
000005CA 050C ; Point to the next ASCII msg
WRITE_DEV_CNTRL:
05CA
43 5F 56 45 44 5F 45 54 49 52 57 00' 05CA
20 2C 4C 52 54 4E 05D6
11 05CA
05DC 413 BUILD RESERVED,00000000
0000050C 05DC ; PC of ASCII pkt RESERVED table
000005DC' 050C ; ASCII pkt address
00000000 0510 ; Expected packet return code
000005DC 0514 ; Point to the next ASCII msg
RESERVED:
05DC
20 2C 44 45 56 52 45 53 45 52 00' 05DC
0A 05DC
05E7 414 BUILD SET_SELF_TEST,00000023
00000514 05E7 ; PC of ASCII pkt SET_SELF_TEST table
000005E7' 0514 ; ASCII pkt address
00000023 0518 ; Expected packet return code
000005E7 051C ; Point to the next ASCII msg
SET_SELF_TEST:
05E7
45 54 5F 46 4C 45 53 5F 54 45 53 00' 05E7
20 2C 54 53 05F3
0F 05E7
05F7 415 BUILD CLR_SELF_TEST,00000003
0000051C 05F7 ; PC of ASCII pkt CLR_SELF_TEST table
000005F7' 051C ; ASCII pkt address
00000003 0520 ; Expected packet return code
000005F7 0524 ; Point to the next ASCII msg
CLR_SELF_TEST:
05F7
45 54 5F 46 4C 45 53 5F 52 4C 43 00' 05F7
20 2C 54 53 0603
0F 05F7
0607 416 BUILD NOOP,00000003
00000524 0607 ; PC of ASCII pkt NOOP table
00000607' 0524 ; ASCII pkt address
00000003 0528 ; Expected packet return code
00000607 052C ; Point to the next ASCII msg
NOOP:
0607
20 2C 50 4F 4F 4E 00' 0607
06 0607
060E 417 BUILD DIAG_READ_INT,00000023
0000052C 060E ; PC of ASCII pkt DIAG_READ_INT table
0000060E' 052C ; ASCII pkt address
00000023 0530 ; Expected packet return code
0000060E 0534 ; Point to the next ASCII msg
DIAG_READ_INT:
060E
49 5F 44 41 45 52 5F 47 41 49 44 00' 060E
20 2C 54 4E 061A
0F 060E
```



00000534	061E	418	BUILD DIAG_WRIT_INT,00000023	
0000061E	0534		.=PC1...	: PC of ASCIC pkt DIAG_WRIT_INT table
00000023	0538		.ADDRESS PC2...	: ASCIC pkt address
0000061E	053C		.LONG ^X00000023	: Expected packet return code
	061E		.=PC2...	: Point to the next ASCIC msg
49 5F 54 49 52 57 5F 47 41 49 44 00	061E		DIAG_WRIT_INT:	
20 2C 54 4E	062A		.ASCIC /DIAG_WRIT_INT, /	
0F	061E			
	062E	419	BUILD READ_DDI,00000023	
0000053C	062E		.=PC1...	: PC of ASCIC pkt READ_DDI table
0000062E	053C		.ADDRESS PC2...	: ASCIC pkt address
00000023	0540		.LONG ^X00000023	: Expected packet return code
0000062E	0544		.=PC2...	: Point to the next ASCIC msg
20 2C 49 44 44 5F 44 41 45 52 00	062E		READ_DDI:	
0A	062E		.ASCIC /READ_DDI, /	
	0639	420	BUILD DIAG_WRT_CNTRL,00000023	
00000544	0639		.=PC1...	: PC of ASCIC pkt DIAG_WRT_CNTRL table
00000639	0544		.ADDRESS PC2...	: ASCIC pkt address
00000023	0548		.LONG ^X00000023	: Expected packet return code
00000639	054C		.=PC2...	: Point to the next ASCIC msg
4E 43 5F 54 52 57 5F 47 41 49 44 00	0639		DIAG_WRT_CNTRL:	
20 2C 4C 52 54	0645		.ASCIC /DIAG_WRT_CNTRL, /	
10	0639			
	064A	421	BUILD SET_RAND_ENABLE,00000000	
0000054C	064A		.=PC1...	: PC of ASCIC pkt SET_RAND_ENABLE table
0000064A	054C		.ADDRESS PC2...	: ASCIC pkt address
00000000	0550		.LONG ^X00000000	: Expected packet return code
0000064A	0554		.=PC2...	: Point to the next ASCIC msg
4E 45 5F 44 4E 41 52 5F 54 45 53 00	064A		SET_RAND_ENABLE:	
20 2C 45 4C 42 41	0656		.ASCIC /SET_RAND_ENABLE, /	
11	064A			
	065C	422	BUILD CLR_RAND_ENABLE,00000000	
00000554	065C		.=PC1...	: PC of ASCIC pkt CLR_RAND_ENABLE table
0000065C	0554		.ADDRESS PC2...	: ASCIC pkt address
00000000	0558		.LONG ^X00000000	: Expected packet return code
0000065C	055C		.=PC2...	: Point to the next ASCIC msg
4E 45 5F 44 4E 41 52 5F 52 4C 43 00	065C		CLR_RAND_ENABLE:	
20 2C 45 4C 42 41	0668		.ASCIC /CLR_RAND_ENABLE, /	
11	065C			
	066E	423	BUILD HALT,00000003	
0000055C	066E		.=PC1...	: PC of ASCIC pkt HALT table
0000066E	055C		.ADDRESS PC2...	: ASCIC pkt address
00000003	0560		.LONG ^X00000003	: Expected packet return code
0000066E	0564		.=PC2...	: Point to the next ASCIC msg
20 2C 54 4C 41 48 00	066E		HALT:	
06	066E		.ASCIC /HALT, /	
	0675	424	BUILD FREE,00000029	
00000564	0675		.=PC1...	: PC of ASCIC pkt FREE table
00000675	0564		.ADDRESS PC2...	: ASCIC pkt address
00000029	0568		.LONG ^X00000029	: Expected packet return code
00000675	056C		.=PC2...	: Point to the next ASCIC msg



UETDR7800  
V04-000

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 I 14  
Read-Only Data 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 13  
5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (3)

20 2C 45 45 52 46 00' 0675 FREE:  
06 0675 .ASCIC /FREE, /  
0675  
067C 425 .NLIST MEB



```
00000000 067C 427 .SBTTL Read/Write Data
00000000 428 .PSECT RWDATA,WRT,NOEXE,PAGE
0000 429
0000 430 TTCHAN: ; Channel associated with ctrl. term.
0000 431 .WORD 0
0002 432
0002 433 MBCHAN: ; Mailbox channel
0000 434 .WORD 0
0004 435
0004 436 CHAN: ; DR780 channel
0000 437 .WORD 0
0006 438
0006 439 PID: ; PID storage for ucode load process
00000000 440 .LONG 0
000A 441
000A 442 FLAG: ; Miscellaneous flag bits
0000 443 .WORD 0 ; (See Equated Symbols for definitions)
000C 444
000C 445 FAO_BUF: ; FAO output string descriptor
0000 0084 446 .WORD TEXT_BUFFER,0
0000001C' 0010 447 .ADDRESS BUFFER
0014 448
0014 449 BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 0084 450 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
0000001C' 0018 451 .ADDRESS BUFFER
001C 452
001C 453 BUFFER: ; FAO output and other misc. buffer
000000A0 001C 454 .BLKB TEXT_BUFFER
00A0 455
00A0 456 DEVDESC: ; Device name descriptor
0000 000A 457 .WORD MAX_DEV_DESIG,0
000000BF' 00A4 458 .ADDRESS DEV_NAME
00A8 459
00A8 460 PROCESS_NAME: ; Process name
38 37 52 44 000000B0'010E0000' 00A8 461 .ASCID /DR78/
0000000B 00B4 462 PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
000000BF 00B4 463 .BLKB PROCESS_NAME_FREE
00BF 464
00BF 465 DEV_NAME: ; Device name buffer
000000CE 00BF 466 .BLKB MAX_DEV_DESIG+MAX_UNIT_DESIG
0000000F 00CE 467 NAME_LEN = .-DEV_NAME
00CE 468
00CE 469 DIB: ; Device Information Block
0000 0074 470 .WORD DIBSK_LENGTH,0
000000D6' 00D2 471 .ADDRESS DIBBUF
00D6 472
0000014A 00D6 473 .BLKB DIBSK_LENGTH
014A 474
014A 475 ERROR_COUNT: ; Cumulative error count at runtime
00000000 014A 476 .LONG 0
014E 477
014E 478 STATUS: ; Status value on program exit
00000000 014E 479 .LONG 0
0152 480
00000000 0152 481 QUAD_STATUS: ; IO status block for misc sys. svcs.
0152 482 .QUAD 0
015A 483
```

00000000	00000000	015A	484	INADDRESS:		; \$CRMPSC address storage
		015A	485	.LONG	0,0	
		0162	486			
00000000	00000000	0162	487	OUTADDRESS:		
		0162	488	.LONG	0,0	
		016A	489			
	0000	016A	490	UNIT_NUMBER:		; Current dev unit number
		016A	491	.WORD	0	
		016C	492			
	0000	016C	493	DEVNAM_LEN:		; Current device name length
		016C	494	.WORD	0	
		016E	495			
AAAAA		016E	496	RANDOM1:		; Random word #1
		016E	497	.LONG	^XAAAAA	
		0172	498			
A72EA72E		0172	499	RANDOM2:		; Random word #2
		0172	500	.LONG	^XA72EA72E	
		0176	501			
00000000		0176	502	ITERATION:		; # of times all tests were executed
		0176	503	.LONG	0	
		017A	504			
00000000		017A	505	PASS:		; Pass count
		017A	506	.LONG	0	
		017E	507			
00000182		017E	508	MSG_BLOCK:		; Auxiliary \$GETMSG info
		017E	509	.BLKB	4	
		0182	510			
00000000		0182	511	EXIT_DESC:		; Exit handler descriptor
00000CF0		0182	512	.LONG	0	
00000001		0186	513	.ADDRESS	EXIT_HANDLER	
0000014E		018A	514	.LONG	1	
		018E	515	.ADDRESS	STATUS	
		0192	516			
00000000		0192	517	ARG_COUNT:		; Argument counter used by ERROR_EXIT
		0192	518	.LONG	0	
		0196	519			
		0196	520	:		
		0196	521	:	Head of self-relative UETP unit block queue.	
		0196	522	:		
		0196	523	:		
		0198	524	.ALIGN	QUAD	
00000000	00000000	0198	525	UNIT_LIST:		; Head of unit block circular list
		0198	526	.QUAD	0	
		01A0	527			
00000000	00000000	01A0	528	NEW_NODE:		; Newly aquired node address
		01A0	529	.QUAD	0	
		01A8	530			
00000000		01A8	531	PKT_CNT:		; Cumulative packet count for this PKT_CHECK
		01A8	532	.LONG	0	
		01AC	533			
00000000		01AC	534	PACK_REMOVED:		; Bit mask record of the packets which
		01AC	535	.LONG	0	; Have been removed from the termination
		01B0	536			; Queue. Bit position is directly related
		01B0	537			; To the fuction code e.g. the READ bit
		01B0	538			; Is 0 and the READ_CHAIN bit is 1
		01B0	539			
		01B0	540	ARGS:		



```
00000200 01B0 541 .BLKL 20 ; Space for 20 arguments
0200 542
0200 543 BADRPKT:
00000000 0200 544 .LONG 0 ; Bad DR packet message desc.
0000001C' 0204 545 .ADDRESS BUFFER
0208 546
0208 547 RATE_BUF:
00000007 0208 548 .LONG 7 ; Buffer for ASCII rate in decimal
00000210' 020C 549 .ADDRESS .+4
00000217 0210 550 .BLKB 7
0217 551
0217 552 RATE_FLOAT:
00000000 00000000 0217 553 .DOUBLE 0 ; Storage for double format rate
021F 554
021F 555 RATE_DESC:
00000000 021F 556 .LONG 0 ; Desc for the full rate message
0000001C' 0223 557 .ADDRESS BUFFER
0227 558
0227 559 DRIostat:
00000000 00000000 0227 560 .QUAD 0 ; DR780 IO status block
022F 561
022F 562 BUFBLK:
022F 563 TEST_DATA:
022F 564 OUTPUT_BUF: ; Primary output data buffer
00000A2F 022F 565 .BLKB BUFSIZ
0A2F 566
0A2F 567 INPUT_BUF: ; Primary input data buffer
0000122F 0A2F 568 .BLKB BUFSIZ
122F 569
122F 570 INPUT1_BUF: ; Secondary input data buffer
000012AF 122F 571 .BLKB 128
00001080 12AF 572 BUFBLSIZ=-BUFBLK
12AF 573
12AF 574 ;
12AF 575 ; Data transfer command table
12AF 576 ;
12AF 577 CMDTBL:
000002E4' 12AF 578 .LONG CMDBLKSIZ ; Length of command block in bytes
000012D8' 12B3 579 .LONG CMDBLK ; Address of start of command block.
12B7 580 ; Also head of input queue
00001080 12B7 581 .LONG BUFBLSIZ ; Length of buffer block in bytes
0000022F' 12BB 582 .LONG BUFBLK ; Base address of the buffer block
00000878' 12BF 583 .LONG PKT1_AST ; Address of the packet interrupt routine
00000000 12C3 584 .LONG 0 ; Parameter to be passed to AST routine.
00 12C7 585 .BYTE 0 ; Data transfer rate
03 12C8 586 .BYTE XF$M_CMT_DIPEAB!-
12C9 587 XF$M_CMT_SETRTE ; Flags byte
0000 12C9 588 .WORD 0 ; Not used
000012CF' 12CB 589 .LONG GOBIT ; Address into which the address of the
12CF 590 ; DR's gobit will be written by QIO.
12CF 591 ;
00000020 12CF 592 CMDTBLSIZ=-CMDTBL ; Define the length of the command table
12CF 593 ;
12CF 594 ; Long word to receive the address of the gobit
12CF 595 ;
12CF 596 GOBIT:
00000000 12CF 597 .LONG 0
```



```
12D3 599 .Subtitle COMMAND BLOCK & PACKETS
12D3 600 :
12D3 601 : This is the start of the command block from which the DR fetches its commands
12D3 602 :
12D3 603 : The commands are in the form of a block of memory called a packet which is
12D3 604 : linked into a list using the interlocked queue instructions.
12D3 605 : The DR removes the command packets from the INPUT QUEUE processes them
12D3 606 : and replaces them onto the TERMINATION QUEUE.
12D3 607 : A status longword is written into each packet before it is connected to
12D3 608 : to the TERMQ.
12D3 609 :
12D3 610 : The command block must be quad word aligned to support the queue instructions
12D3 611 :
00'00'00'00'00' 12D3 612 .ALIGN QUAD 0
12D8 613 CMDBLK:: ;
000012E0 12D8 614 INPTQH: .BLKQ 1
000012E8 12E0 615 TERMQH: .BLKQ 1
000012F0 12E8 616 FREEQH: .BLKQ 1
12F0 617 :
12F0 618 : Packet to do a nop command.
12F0 619 :
12F0 620 NOOP_PKT:
00000000 12F0 621 .LONG 0 ; Queue forward link
00000000 12F4 622 .LONG 0 ; Queue backward link
0000 12F8 623 .WORD 0 ; Log area and message length
08 12FA 624 .BYTE XF$K_PKT_NOP@XF$V_PKT_FUNC ; Command = nop
80 12FB 625 .BYTE XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
12FC 626 ; Interrupt unconditionally.
00000000 12FC 627 .LONG 0 ; Byte count not used here
00000000 1300 628 .LONG 0 ; Va not used here
00000000 1304 629 .LONG 0 ; Residual memory byte count
00000000 1308 630 ; Not used here.
00000000 1308 631 .LONG 0 ; Residual ddi byte count
130C 632 ; Not used here.
00000000 130C 633 .LONG 0 ; Dr status longword for this pkt
1310 634 ;
1310 635 :
1310 636 : Packet to do a halt command.
1310 637 :
1310 638 : This packet will cause two AST's to be queued regardless of the state
1310 639 : of the interrupt control field.
1310 640 :
1310 641 :
1310 642 :
1310 643 HALT_PKT:
00000000 1310 644 .LONG 0 ; Queue forward link
00000000 1314 645 .LONG 0 ; Queue backward link
0000 1318 646 .WORD 0 ; Log area and message length
000F 131A 647 .WORD XF$K_PKT_HALT@XF$V_PKT_FUNC ; Command = halt
131C 648 ; Interrupt field ignored here
00000000 131C 649 .LONG 0 ; Byte count not used here
00000000 1320 650 .LONG 0 ; Va not used here
00000000 1324 651 .LONG 0 ; Residual memory byte count
1328 652 ; Not used here.
00000000 1328 653 .LONG 0 ; Residual DDI byte count
132C 654 ; Not used here.
00000000 132C 655 .LONG 0 ; DR status longword for this pkt
```



```

1330 656 ;
1330 657 ;
1330 658 : Packet to set self test mode.
1330 659 :
1330 660 : This packet will get error status if DDI DISABLE is set when it is executed.
1330 661 :
1330 662 SET_SELF_PKT:
1330 663 :
00000000 1330 664 .LONG 0 : Queue forward link
00000000 1334 665 .LONG 0 : Queue backward link
0000 1338 666 .WORD 0 : Log area and message length
06 133A 667 .BYTE XF$K_PKT_SETTST@XF$V_PKT_FUNC ; Command = set self test
80 133B 668 .BYTE XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
00000000 133C 669 .LONG 0 : Byte count not used here
00000000 1340 670 .LONG 0 : Va not used here
00000000 1344 671 .LONG 0 : Residual memory byte count
1348 672 : Not used here.
00000000 1348 673 .LONG 0 : Residual DDI byte count
134C 674 : Not used here.
00000000 134C 675 .LONG 0 : DR status longword for this pkt
1350 676 :
1350 677 :
1350 678 : Packet to clear self test.
1350 679 :
1350 680 :
1350 681 CLR_SELF_PKT:
1350 682 :
00000000 1350 683 .LONG 0 : Queue forward link
00000000 1354 684 .LONG 0 : Queue backward link
0000 1358 685 .WORD 0 : Log area and message length
07 135A 686 .BYTE XF$K_PKT_CLRTST@XF$V_PKT_FUNC ; Command = clear self test
80 135B 687 .BYTE XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
00000000 135C 688 .LONG 0 : Byte count not used here
00000000 1360 689 .LONG 0 : Va not used here
00000000 1364 690 .LONG 0 : Residual memory byte count
1368 691 : Not used here.
00000000 1368 692 .LONG 0 : Residual DDI byte count
136C 693 : Not used here.
00000000 136C 694 .LONG 0 : DR status longword for this pkt
1370 695 :
1370 696 :
1370 697 : Command packet to do a diagnostic write internal
1370 698 :
1370 699 : This command is used to test the dr's internal silo.
1370 700 : The number of bytes specified by the byte count are read from memory
1370 701 : and stored in the silo.
1370 702 :
1370 703 :
1370 704 DIAG_WRI_PKT:
1370 705 :
00000000 1370 706 .LONG 0 : Queue forward link
00000000 1374 707 .LONG 0 : Queue backward link
0000 1378 708 .WORD 0 : Log area and message length
0A 137A 709 .BYTE XF$K_PKT_DIAGWI@XF$V_PKT_FUNC ; Command = diag write internal
80 137B 710 .BYTE XF$K_PKT_NOINT@XF$V_PKT_INTCTL ; No interrupt.
00000080 137C 711 .LONG 128 : Byte count is 128 even though
1380 712 : Only 124 bytes are valid

```



```
0000022F' 1380 713 .ADDRESS OUTPUT_BUF ; Address of data buffer
00000000 1384 714 .LONG 0 ; Residual memory byte count
00000000 1388 715 .LONG 0 ; Residual DDI byte count
00000000 138C 716 .LONG 0 ; DR status long word for this pkt
1390 717 ;
1390 718 ;
1390 719 : Command packet to do a diagnostic read internal command.
1390 720 :
1390 721 : This command is used to read the data in the DR's internal silo.
1390 722 : The number of bytes specified by the byte count are read from the dr
1390 723 : and written to memory specified by the virtual address field.
1390 724 :
1390 725 :
1390 726 :
1390 727 DIAG_REA_PKT: ;
00000000 1390 728 .LONG 0 ; Queue forward link
00000000 1394 729 .LONG 0 ; Queue backward link
0000 1398 730 .WORD 0 ; Log area and message length
09 139A 731 .BYTE XF$K_PKT_DIAGRIA$X$V_PKT_FUNC ; Command = diag read internal
00 139B 732 .BYTE XF$K_PKT_UNCONDA$X$V_PKT_INTCTL ; Interrupt unconditionally
00000080 139C 733 .LONG 128 ; Byte count is 128 even though
13A0 734 ; Only 124 bytes are valid
0000122F' 13A0 735 .ADDRESS INPUT1_BUF ; Address of data buffer
00000000 13A4 736 .LONG 0 ; Residual memory byte count
00000000 13A8 737 .LONG 0 ; Residual DDI byte count
00000000 13AC 738 .LONG 0 ; DR status long word for this pkt
1380 739 ;
1380 740 :
1380 741 : Command packet to do a diagnostic read DDI command
1380 742 :
1380 743 : This command wraps the data around on the DDI bus and stores it back into
1380 744 : the silo.
1380 745 :
1380 746 :
1380 747 READ_DDI_PKT: ;
1380 748 ;
00000000 1380 749 .LONG 0 ; Queue forward link
00000000 1384 750 .LONG 0 ; Queue backward link
0000 1388 751 .WORD 0 ; Log area and message length
0B 138A 752 .BYTE XF$K_PKT_DIAGRD$X$V_PKT_FUNC ; Command = diag read DDI
80 138B 753 .BYTE XF$K_PKT_NOINT$X$V_PKT_INTCTL ; No interrupt.
00000080 138C 754 .LONG 128 ; Byte count = silo size
00000000 13C0 755 .LONG 0 ; Address field not used
00000000 13C4 756 .LONG 0 ; Residual memory byte count
00000000 13C8 757 .LONG 0 ; Residual DDI byte count
00000000 13CC 758 .LONG 0 ; DR status long word for this pkt
13D0 759 ;
13D0 760 :
13D0 761 : Packet to do a write chained command.
13D0 762 :
13D0 763 WRITE_CH_PKT: ;
00000000 13D0 764 .LONG 0 ; Queue forward link
00000000 13D4 765 .LONG 0 ; Queue backward link
0010 13D8 766 .WORD 16 ; Log area and message length
03 13DA 767 .BYTE XF$K_PKT_WRTCHN$X$V_PKT_FUNC ; Command = write chained
98 13DB 768 .BYTE <XF$R_PKT_CBDMBC$X$V_PKT_CISEL>!--
13DC 769 <XF$K_PKT_NOINT$X$V_PKT_INTCTL> ; No interrupt. Send command,
```



```
13DC 770 ; Byte count,device message
13DC 771 ; Message is 16 bytes long
0000003B 13DC 772 .LONG 59 ; Byte count is 59 to keep
13E0 773 ; Things on odd boundries.
0000022F' 13E0 774 .ADDRESS OUTPUT_BUF ; Address of data buffer
00000000 13E4 775 .LONG 0 ; Residual memory byte count
00000000 13E8 776 .LONG 0 ; Residual DDI byte count
00000000 13EC 777 .LONG 0 ; DR status long word for this pkt
13F0 778 ; Generate and incrementing pattern
13F0 779 ; For the device message.
00000000 13F0 780 X=0
13F0 781 .REPT 16 ; Device message is 16 bytes long
13F0 782 .BYTE X
13F0 783 X=X+1
00 13F0 784 .ENDR
1400 785 ;
1400 786 ;
1400 787 ; Command packet do to a write device command
1400 788 ;
1400 789 WRITE_PKT:
00000000 1400 790 .LONG 0 ; Queue forward link
00000000 1404 791 .LONG 0 ; Queue backward link
0080 1408 792 .WORD 128 ; Log area and message length
02 140A 793 .BYTE XF$K_PKT_WRT@XF$V_PKT_FUNC ; Command = write device
98 140B 794 .BYTE <XF$R_PKT_CBDMBC@XF$V_PKT_CISEL>!-
140C 795 <XF$K_PKT_NOINT@XF$V_PKT_INTCTL> ; No interrupt. Send command.
140C 796 ; Byte count,device message
140C 797 ; Message is 128 bytes long
000007C5 140C 798 .LONG 1989 ; Byte count is 1989 to keep
1410 799 ; Things on odd boundries.
0000026A' 1410 800 .ADDRESS OUTPUT_BUF+59 ; Address of data buffer
00000000 1414 801 .LONG 0 ; Residual memory byte count
00000000 1418 802 .LONG 0 ; Residual DDI byte count
00000000 141C 803 .LONG 0 ; DR status long word for this pkt
1420 804 ; Device message for this packet
1420 805 ; Even though in self test mode
1420 806 ; The dr will not look at the message
000000FF 1420 807 ; An incrementing pattern is used.
1420 808 X=^XFF
1420 809 .REPT 128 ; Generate an decrementing pattern
1420 810 .BYTE X
1420 811 X=X-1
FF 1420 812 .ENDR
14A0 813 ;
14A0 814 ;
14A0 815 ; Command packet to do a read chained command.
14A0 816 ;
14A0 817 ; This packet must only be executed in self test mode.
14A0 818 ; A device message is transmitted to never never land to use more
14A0 819 ; microcode in the DR.
14A0 820 ;
00000000 14A0 821 READ_CHA_PKT:
00000000 14A0 822 .LONG 0 ; Queue forward link
0010 14A4 823 .LONG 0 ; Queue backward link
01 14A8 824 .WORD 16 ; Log area and message length
98 14AA 825 .BYTE XF$K_PKT_RDCHN@XF$V_PKT_FUNC ; Command = read chained
14AB 826 .BYTE <XF$R_PKT_CBDMBC@XF$V_PRT_CISEL>!-
```



```
14AC 827 <XFSK_PKT_NOINT@XFSV_PKT_INTCTL> ; No interrupt device message
14AC 828 ; 16 bytes long, send command
14AC 829 ; Byte count and device message
14AC 830
0000003B 14AC 831 .LONG 59 ; Byte count is 59 to keep
14B0 832 ; Things on odd boundaries.
00000A2F' 14B0 833 .ADDRESS INPUT_BUF ; Address of data buffer
00000000 14B4 834 .LONG 0 ; Residual memory byte count
00000000 14B8 835 .LONG 0 ; Residual DDI byte count
00000000 14BC 836 .LONG 0 ; DR status long word for this pkt
14C0 837 ; Generate and incrementing pattern
14C0 838 ; For the device message.
00000000 14C0 839 X=0
14C0 840 .REPT 16 ; Device message is 16 bytes long
14C0 841 .BYTE X
14C0 842 X=X+1
00 14C0 843 .ENDR
14D0 844 ;
14D0 845 ;
14D0 846 ; Command packet do to a read device command
14D0 847 ;
14D0 848 READ_PKT:
00000000 14D0 849 .LONG 0 ; Queue forward link
00000000 14D4 850 .LONG 0 ; Queue backward link
0080 14D8 851 .WORD 128 ; Log area and message length
00 14DA 852 .BYTE XFSK_PKT_RD@XFSV_PKT_FUNC ; Command = read device
18 14DB 853 .BYTE <XFSK_PKT_CBDMBC@XFSV_PKT_CISEL>!--
14DC 854 <XFSK_PKT_UNCOND@XFSV_PKT_INTCTL> ; Interrupt when done, send command
14DC 855 ; Byte count and device message.
14DC 856 ; Device message is 128 bytes.
000007C5 14DC 857 .LONG 1989 ; Byte count is 1989 to keep
14E0 858 ; Things on odd boundaries.
00000A6A' 14E0 859 .ADDRESS INPUT_BUF+59 ; Address of data buffer
00000000 14E4 860 .LONG 0 ; Residual memory byte count
00000000 14E8 861 .LONG 0 ; Residual DDI byte count
00000000 14EC 862 .LONG 0 ; DR status long word for this pkt
14F0 863 ; Device message for this packet
14F0 864 ; Even though in self test mode
14F0 865 ; The dr will not look at the message
14F0 866 ; An incrementing pattern is used.
00000000 14F0 867 X=0
14F0 868 .REPT 128 ; Generate an incrementing pattern
14F0 869 .BYTE X
14F0 870 X=X+1
00 14F0 871 .ENDR
1570 872 ;
1570 873 ;
1570 874 ; Command packet to do a diagnostic write device message command.
1570 875 ;
1570 876 ; This command writes a single byte onto the control bus and reads it
1570 877 ; back again. A packet is then removed from FREE Q and the data read is
1570 878 ; placed into the message area of this packet.
1570 879 ;
1570 880 ; This command can only be executed in self test mode.
1570 881 ;
1570 882 DIAG_WRT_PKT:
1570 883 ;
```



```
00000000 1570 884 .LONG 0 ; Queue forward link
00000000 1574 885 .LONG 0 ; Queue backward link
0001 1578 886 .WORD 1 ; Log area and message length
000C 157A 887 .WORD XFSK_PKT_DIAGWC@XFSV_PKT_FUNC ; Command = diagnostic write
80 157C 888 .BYTE XFSK_PKT_NOINT@XFSV_PKT_INTCTL ; No interrupt.
157D 889
00000000 157D 890 .LONG 0 ; Byte count not used here
00000000 1581 891 .LONG 0 ; Va field not used here
00000000 1585 892 .LONG 0 ; Residual memory byte count
1589 893 ; Not used here
00000000 1589 894 .LONG 0 ; Residual DDI byte count
158D 895 ; Not used here.
00000000 158D 896 .LONG 0 ; DR status longword for this pkt.
1591 897 DIAG_CNTRL_MESS:
000000AA 1591 898 .LONG ^X0AA ; Longword for the device message
1595 899 ; This is modified dynamically
1595 900
1595 901 ;
1595 902 ; Packet to be placed onto FREE QUEUE to receive the message from the
1595 903 ; diagnostic write device message command.
1595 904 ;
1595 905 ;
1595 906 .ALIGN QUAD
1598 907 FREE_PKT:
1598 908
00000000 1598 909 .LONG 0 ; Queue forward link
00000000 159C 910 .LONG 0 ; Queue backward link
00000001 15A0 911 .LONG 1 ; Reserve 1 byte for the incoming
15A4 912 ; Message
00000000 15A4 913 .LONG 0 ; Byte count not used here
00000000 15A8 914 .LONG 0 ; Va not used here
00000000 15AC 915 .LONG 0 ; Residual byte counts not used here
00000000 15B0 916 .LONG 0
00000000 15B4 917 .LONG 0 ; DR status longword for this packet.
00000000 15B8 918 .LONG 0 ; Long word to receive the message byte.
15BC 919
15BC 920
15BC 921 ;
15BC 922 ; End of command block
15BC 923 ;
000002E4 15BC 924 CMDBLKSIZE=-CMDBLK ; Define the length of the
15BC 925 CMDBLKEND: ; Command block
```



```

15BC 927      .SBTTL RMS-32 Data Structures
15BC 928      .ALIGN LONG
15BC 929
15BC 930 SYSIN_FAB:                                ; Allocate FAB for SYSS$INPUT
15BC 931      $FAB-
15BC 932      FNM = <SYSS$INPUT>
160C 933
160C 934 SYSIN_RAB:                                ; Allocate RAB for SYSS$INPUT
160C 935      $RAB-
160C 936      FAB = SYSIN_FAB,-
160C 937      ROP = PMT,-
160C 938      PBF = PROMPT,-
160C 939      PSZ = PMTSIZ,-
160C 940      UBF = DEV_NAME,-
160C 941      USZ = NAME_LEN
1650 942
1650 943 INI_FAB:                                    ; Allocate FAB for UETINIDEV
1650 944      $FAB-
1650 945      FAC = <GET,PUT,UPD>,-
1650 946      RAT = CR,-
1650 947      SHR = <GET,PUT,UPI>,-
1650 948      FNM = <UETINIDEV.DAT>
16A0 949
16A0 950 INI_RAB:                                    ; Allocate RAB for UETINIDEV
16A0 951      $RAB-
16A0 952      FAB = INI_FAB,-
16A0 953      RBF = BUFFER,-
16A0 954      UBF = BUFFER,-
16A0 955      USZ = REC_SIZE
16E4 956
000016EA 16E4 957 DDB_RFA:                            ; RFA storage for INI_RAB
16E4 958      .BLKB 6
16EA 959
16EA 960      .ALIGN LONG
16EC 961 SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
16EC 962      $FAB-
16EC 963      FAC = GET,-
16EC 964      SHR = <UPI,GET>,-
16EC 965      RAT = CR,-
16EC 966      FOP = UFO,-
16EC 967      FNM = <UETSUPDEV.DAT>
173C 968
173C 969 ;
173C 970 ; Dummy FAB and RAB to copy to the UETP unit blocks
173C 971 ; The following FAB and RAB must be contiguous and in this order!
173C 972 ;
173C 973
173C 974 DUMMY_FAB:
173C 975      $FAB
178C 976
178C 977 DUMMY_RAB:
178C 978      $RAB      RSZ = WRITE_SIZE,-
178C 979      USZ = READ_SIZE
17D0 980
17D0 981 XFLDR_SYSS$ERROR_FAB:                            ; Gets possible log file from ucode ldr
17D0 982      $FAB      FNS = XFLDR_SYSS$ERROR_LENGTH,-
17D0 983      FNA = XFLDR_SYSS$ERROR

```



UETDR7800  
V04-000

G 15

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 24  
RMS-32 Data Structures 5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (5)

1820	984	
1820	985	XFLDR_SYSSERROR_RAB:
1820	986	SRAB FAB = XFLDR_SYSSERROR_FAB,-
1820	987	USZ = TEXT_BUFFER,-
1820	988	UBF = BUFFER



```
1864 990 .SBTTL Main Program
00000000 991 .PSECT DR78,EXE,NOWRT,PAGE
0000 992
0000 993 .DEFAULT DISPLACEMENT,WORD
0000 994
0000 995
0000 996 :+ This is the main code for the DR780 test. The byte transfer sizes were chosen
0000 997 : for hardware dependent reasons. The test sequence is as follows:
0000 998
0000 999 1. do a NOP packet
0000 1000 2. do a set self test mode packet
0000 1001 3. do a 128. byte write diagnostic internal packet
0000 1002 4. do a 128. byte read diagnostic DDI packet
0000 1003 5. do a 128. byte read diagnostic internal packet
0000 1004 6. do a 59. byte write chained packet
0000 1005 7. do a 1989. byte write packet
0000 1006 8. do a 59. byte read chained packet
0000 1007 9. do a 1989. byte read packet
0000 1008 10. do a 1. byte diagnostic control message packet
0000 1009 11. do a halt packet
0000 1010 12. set the go bit
0000 1011 13. check the 124. bytes transfered by the diag write/read internal
0000 1012 14. check the packet status and packet count
0000 1013 15. check for control C's
0000 1014 16. loop until 3 minutes are up
0000 1015
0000 1016 :-
0000 1017
0000 1018 .ENTRY UETDR7800,^M<> ; Entry mask
0002 1019
6D 0AE7'CF DE 0002 1020 MOVAL SSERROR,(FP) ; Declare exception handler
0007 1021 $SETSFMS ENBFLG = #1 ; Enable system service failure mode
0010 1022 $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
001B 1023
001B 1024 $OPEN FAB = SYSIN FAB,- ; Open SYS$INPUT
001B 1025 ERR = RMS_ERROR
002A 1026 $CONNECT RAB = SYSIN RAB,- ; Connect RAB to SYS$INPUT
002A 1027 ERR = RMS_ERROR
02 02 E1 0039 1028 BBC S^#DEV$V TRM,- ; BR if SYS$INPUT is NOT a terminal
1E 15FC'CF 003B 1029 SYSIN FAB+FAB$S DEV,10$
003F 1030 $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
003F 1031 RSLLEN = DEVNAM_LEN,- ; ...a logical name...
003F 1032 RSLBUF = DEVDESC ; ...for the controller to test
01 50 D1 0058 1033 CMPL RO,#SS$ NORMAL ; Was a controller specified?
2E 13 005B 1034 BEQL PROC_CONT_NAME ; BR if it was - go process it
005D 1035 10$:
005D 1036 $GET RAB = SYSIN RAB,- ; Read SYS$INPUT...
005D 1037 ERR = RMS_ERROR ; ...for the controller name
162E'CF B0 006C 1038 MOVW SYSIN RAB+RAB$W_RSZ,- ; Save the name length
016C'CF 0070 1039 DEVNAM_LEN
16 12 0073 1040 BNEQ PROC_CONT_NAME ; BR if we got something
014E'CF 14 D0 0075 1041 MOVL #SS$-BADPARAM,STATUS ; Save an exit status if not
02D5'CF DF 007A 1042 PUSHAL NO_CTRLNAME ; Prepare for message...
01 DD 007E 1043 PUSHL #1 ; ...arg count
00741132 8F DD 0080 1044 PUSHL #UETP$_TEXT!ST$K_ERROR ; ...signal name
03 DD 0086 1045 PUSHL #3 ; ...arg count
OBE5 31 0088 1046 BRW ERROR_EXIT ; ...go tell of bad setup
```



```
00A0'CF 016C'CF 3C 008B 1047
00A0'CF 00A0'CF DF 008B 1048 PROC_CONT NAME:
00A0'CF 00A0'CF DF 008B 1049 MOVZWL DEVDSC ; Set the device name length
00000000'GF 02 FB 0092 1050 PUSHAL DEVDSC ; Make sure...
52 00A0'CF 01 C1 0096 1051 PUSHAL DEVDSC ; ...that the specified controller...
00A8'CF 52 A0 009A 1052 CALLS #2,G^STR$UPCASE ; ...is all uppercase for later comaparison
50 00B4'CF 0B C3 00A1 1053 ADDL3 #1,DEVDS R2 ; Estimate the eventual...
51 52 15 00A7 1054 ADDW2 R2,PROCESS NAME ; ...process name length (incl. "'")
50 51 C2 00AC 1055 MOVAL PROCESS NAME+8- ; Locate first available byte...
00A8'CF 0F B0 00AD 1056 +MAX PROC NAME- ; ...in process name handle...
00B4'CF 0B C3 00AD 1057 -PROCESS NAME FREE,R0 ; ...for device name
51 52 15 00B1 1058 SUBL3 #PROCESS NAME FREE,- ; Will the device name fit...
50 51 C2 00B3 1059 BLEQ R2,R1 ; ...in the remaining space?
00A8'CF 0F B0 00B5 1060 SUBL2 10$ ; BR if it will
00B4'CF 0B C3 00B7 1061 MOVW R1,R0 ; Overwrite handle otherwise...
60 00BF'CF 80 5F 8F 90 00BA 1062 #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
00A0'CF 00A0'CF 28 00BF 1063 MOVW #A/ /,(R0)+ ; Separate handle from device name
000F'CF 7E D4 00BF 1064 MOVW DEVDS,DEV_NAME,(R0) ; Concatenate handle with device name
00741039 8F DD 00CB 1065 CLRL -(SP) ; Set the time stamp flag
00000000'GF 04 FB 00CD 1066 PUSHAL TEST_NAME ; Set the test name
00A8'CF 20 A8 00D1 1067 PUSHL #2 ; Push the argument count
00741039 8F DD 00D3 1068 PUSHL #UETP$ BEGIN!STSSK_SUCCESS ; Set the message code
00000000'GF 04 FB 00D9 1069 CALLS #4,G^LIB$SIGNAL ; Print the startup message
00A8'CF 20 A8 00E0 1070 BISW2 #BEGIN MSGM,FLAG ; Set flag so we don't print it again
00E5 1071 $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDR7800_x
00F0 1072 BBC S^#DEV$V TRM,- ; BR if SYSS$INPUT is NOT a terminal
00F0 1073 SYSIN FAB+FAB$ DEV,20$ ; Get the name of...
66 15FC'CF 02 E1 00F2 1074 $GETDVI_S DEVDSC = SYSS$INPUT,- ; ...device which may abort test
00F6 1075 EFN = #SS SYNCH EFN,- ; ITMLST = INPUT ITMLST,-
00F6 1076 IOSB = QUAD STATUS
00F6 1077 BLBC QUAD STATUS,20$ ; Avoid CTRL/C handler if any error
45 0152'CF 02 E9 00F6 1078 $ASSIGN_S DEVDSC = BUFFER_PTR,- ; Set up for CTRL/C AST handler
0112 1079 CHAN = TTCHAN
0117 1080 $QIOW_S CHAN = TTCHAN,- ; Enable CTRL/C AST's...
0117 1081 FUNC = #IOS SETMODE!IOSM_CTRLCAST,-
0128 1082 P1 = CCASTHAND
0128 1083 PUSHAL PROCESS_NAME ; ...and tell the user...
0128 1084 PUSHL #1
0128 1085 PUSHL #UETP$ ABORTC!STSSK_SUCCESS ; ...how to abort gracefully...
00A8'CF 01 DD 014D 1086 CALLS #3,G^LIB$SIGNAL ; ...
0074832B 8F DD 014F 1087
00000000'GF 03 FB 0155 1088
015C 1089
015C 1090 20$:
015C 1091
```



```
015C 1093 :  
015C 1094 : From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller  
015C 1095 : and unit configuration and lets us know if the setup to run this test was  
015C 1096 : done correctly.  
015C 1097 :  
015C 1098 $OPEN FAB = INI_FAB,- ; Open file 'UETINIDEV.DAT'  
015C 1099 ERR = RMS_ERROR  
016B 1100 $CONNECT RAB = INI_RAB,- ; Connect the RAB and FAB  
016B 1101 ERR = RMS_ERROR  
017A 1102 $MGBLSC_S INADR = INADDRESS,- ; Connect to UETSUPDEV global section  
017A 1103 RETADR = OUTADDRESS,-  
017A 1104 GSDNAM = SUPDEV_GBLSEC,-  
017A 1105 FLAGS = #SEC$M_EXPREG  
00000978 8F 50 D1 0199 1106 CMPL R0,#SS$NOSUCHSEC ; Was the section already there?  
37 12 01A0 1107 BNEQ 30$ ; BR if it was...  
01A2 1108 $OPEN FAB = SUP_FAB,- ; ...else open 'UETSUPDEV.DAT'  
01A2 1109 ERR = RMS_ERROR  
01B1 1110 $CRMPSC_S CHAN = SUP_FAB+FAB$STV,- ; Create the global section  
01B1 1111 INADR = INADDRESS,-  
01B1 1112 RETADR = OUTADDRESS,-  
01B1 1113 GSDNAM = SUPDEV_GBLSEC,-  
01B1 1114 FLAGS = #SEC$M_EXPREG!SEC$M_GBL  
56 0166'CF 0162'CF C3 01D9 1115 30$: SUBL3 OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length  
01E1 1116  
01E1 1117 FIND_IT:  
01E1 1118 $GET RAB = INI_RAB,- ; Get the first record  
01E1 1119 ERR = RMS_ERROR  
01F0 1121 PUSHAL CONT_DESC ; Make sure...  
0416'CF DF 01F4 1122 PUSHAL CONT_DESC ; ...that the controller name...  
0416'CF DF 01F8 1123 CALLS #2,G*STR$UPCASE ; ...is all uppercase letters  
00000000'GF 02 FB 01FF 1124 CMPB #^A/D/,BUFFER ; Is this a DDB?  
001C'CF 44 8F 91 0205 1125 BEQL 10$ ; Go on if not  
27 13 0207 1126 CMPB #^A/E/,BUFFER ; Is this the end of the file?  
001C'CF 45 8F 91 020D 1127 BNEQ FIND_IT ; Continue on if not  
D2 12 020F 1128 PUSHAL DEV$DC ; Push device not supported message  
00A0'CF DF 0213 1129 PUSHAL PROCESS_NAME ; Parameters on the stack  
00A8'CF DF 0217 1130 PUSHAL #2  
02 02 0219 1131 PUSHAL #UETP$DENOSU  
00748333 8F DD 021F 1132 INSV #ST$K_ERROR,- ; Set the severity code...  
02 02 0221 1133 #ST$V_SEVERITY,-  
00 02 0222 1134 #ST$S_SEVERITY,(SP)  
6E 03 0224 1135 MOVL (SP),STATUS ; ...and save it as the exit status  
014E'CF 6E DD 0229 1136 PUSHAL #4  
04 DD 022B 1137 BRW ERROR_EXIT ; Exit in error  
0A42 31 022E 1138 10$: CMPC DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?  
00BF'CF 0022'CF 016C'CF 29 022E 1139 BNEQ FIND_IT ; BR if not  
A7 12 0238 1140 MOVC3 #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address  
16E4'CF 16B0'CF 06 28 023A 1141 CMPB #^A/T/,BUFFER+4 ; Can we test this controller?  
0020'CF 54 8F 91 0242 1142 BEQL FOUND_IT ; BR if we can...  
2F 13 0248 1143 $FAO_S CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't  
024A 1144 OUTLEN = BUFFER_PTR,-  
024A 1145 OUTBUF = FAO_BUF,-  
024A 1146 P1 = #DEV$DC  
014E'CF 14 DD 0263 1148 MOVL #SS$BADPARAM,STATUS ; Set return status  
0014'CF DF 0268 1149 PUSHAL BUFFER_PTR ; ...
```



```
00741132 01 DD 026C 1150      PUSHL #1
00741132 03 DD 026E 1151      PUSHL #UETPS_TEXT!STSSK_ERROR
00741132 09F7 31 0274 1152      PUSHL #3
00741132 09F7 31 0276 1153      BRW ERROR_EXIT
                                ; We can't test what we can't test

                                FOUND_IT:
00741132 09F7 31 0279 1154      $GET RAB = INI_RAB,-
                                ; Get a record
00741132 09F7 31 0279 1155      ERR = RMS_ERROR
00741132 09F7 31 0279 1156      PUSHAL CONT_DESC
                                ; Make sure...
00741132 09F7 31 0279 1157      PUSHAL CONT_DESC
                                ; ...that this line...
00741132 09F7 31 0288 1158      CALLS #2,G*STR$UPCASE
                                ; ...is all uppercase letters
00741132 09F7 31 028C 1159      CMPB #^A/U/,BUFFER
                                ; Is this a UCB?
00741132 09F7 31 0290 1160      BEQL 30$
                                ; BR if it is
00741132 09F7 31 0297 1161      CMPB #^A/D/,BUFFER
                                ; Is this a DDB?
00741132 09F7 31 029D 1162      BEQL 20$
                                ; BR if yes
00741132 09F7 31 029F 1163      CMPB #^A/E/,BUFFER
                                ; Is this the end?
00741132 09F7 31 02A5 1164      BEQL 20$
                                ; BR if yes
00741132 09F7 31 02A7 1165      BEQL 20$
                                ; BR if yes
00741132 09F7 31 02AD 1166      10$:
00741132 09F7 31 02AF 1167      PUSHAL ILLEGAL_REC
                                ; Then this is an error in the record
00741132 09F7 31 02B3 1168      PUSHL #1
                                ; Push the error message
00741132 09F7 31 02B5 1169      PUSHL #UETPS_TEXT!STSSK_ERROR
                                ; Push the signal name
00741132 09F7 31 02BB 1171      PUSHL #3
                                ; Push the temp arg count
00741132 09F7 31 02BD 1172      BRW ERROR_EXIT
                                ; Finish for good
00741132 09F7 31 02C0 1173      20$:
00741132 09F7 31 02C0 1174      BRW ALL_SET
                                ; Found DDB or END
00741132 09F7 31 02C3 1175      30$:
00741132 09F7 31 02C3 1176      CMPB #^A/T/,BUFFER+4
                                ; Is the unit testable?
00741132 09F7 31 02C9 1177      BNEQ FOUND_IT
                                ; BR if not
00741132 09F7 31 02CB 1178      PUSHL #1
                                ; Flag to ignore blanks when converting
00741132 09F7 31 02CD 1179      PUSHL #2
                                ; Set byte size of results
00741132 09F7 31 02CF 1180      PUSHAL UNIT_NUMBER
                                ; Set address to receive word
00741132 09F7 31 02D3 1181      PUSHAL UNIT_DESC
                                ; Push string address
00741132 09F7 31 02D7 1182      CALLS #4,G*OT$SCVT_TI_L
                                ; Convert ASCII unit # to decimal
00741132 09F7 31 02DE 1183      BLBC R0,10$
                                ; Don't allow bogus unit to pass
00741132 09F7 31 02E1 1184      SKPC #^A/ /,MAX_UNIT_DESIG,-
                                ; Find out where unit number really is
00741132 09F7 31 02E4 1185      BUFFER+6
00741132 09F7 31 02E7 1186      DECL R0
                                ; Units must all be at least one digit
00741132 09F7 31 02E9 1187      SKPC #^A/O/,R0,(R1)
                                ; Skip leading zeroes on the unit
00741132 09F7 31 02ED 1188      INCL R0
                                ; Compensate for DECL above
00741132 09F7 31 02EF 1189      ADDW3 R0,DEVNAM_LEN,DEVVSC
                                ; Calculate device unit string length
00741132 09F7 31 02F7 1190      MOVZWL DEVNAM_LEN,R2
                                ; Offset to unit number in DEVVSC
00741132 09F7 31 02FC 1191      MOVV3 R0,(R1),DEV_NAME(R2)
                                ; Append unit number to device
00741132 09F7 31 0302 1192      $GETDEV_S DEVNAM = DEVVSC,-
                                ; Get the device characteristics
00741132 09F7 31 0302 1193      PRIBUF = DIB
00741132 09F7 31 0317 1194      MOVZBL DIBBUF+DIB$B_DEVCLASS,R7
                                ; Save the device class
00741132 09F7 31 031C 1195      MOVZBL DIBBUF+DIB$B_DEVTYPE,R8
                                ; Save the device type
00741132 09F7 31 0321 1196      $FAO_S CTRSTR = CS1,-
00741132 09F7 31 0321 1197      OUTBUF = FAO_BUF,-
00741132 09F7 31 0321 1198      P1 = R7,-
00741132 09F7 31 0321 1199      P2 = R8
                                ; Make it into a string
00741132 09F7 31 0336 1200      MATCHC #6,BUFFER,R6,@OUTADDRESS
                                ; Find the device class and type
00741132 09F7 31 033F 1201      BEQL 40$
                                ; BR if it was found
00741132 09F7 31 0341 1202      $FAO_S CTRSTR = CS3,-
                                ; Try for full class support
00741132 09F7 31 0341 1203      OUTBUF = FAO_BUF,-
00741132 09F7 31 0341 1204      P1 = R7
00741132 09F7 31 0354 1205      MATCHC #6,BUFFER,R6,@OUTADDRESS
                                ; Find the device class only
00741132 09F7 31 035D 1206      BNEQ 50$
                                ; BR if not found
```



55	000F'CF	9A	035F	1207	40\$:			
0017'CF	63	55	29	035F	1208	MOVZBL	TEST_NAME,R5	; Get the test name length
	1F	13	0364	1209		CMPC3	R5,(R3),TEST_NAME+8	; Are we the right test?
			036A	1210		BEQL	60\$	; BR if yes
			036C	1211	50\$:			
	00A0'CF	DF	036C	1212		PUSHAL	DEVDS	; Push device not supported message
	00A8'CF	DF	0370	1213		PUSHAL	PROCESS_NAME	; Parameters on the stack
	02	DD	0374	1214		PUSHL	#2	; Push the argument count
00748333	8F	DD	0376	1215		PUSHL	#UETPS_DENOSU	
	02	FO	037C	1216		INSV	#STSSK_ERROR,-	
	00		037E	1217			#STSSV_SEVERITY,-	
6E	03		037F	1218			#STSSS_SEVERITY,(SP)	; Set the severity code...
014E'CF	6E	DO	0381	1219		MOVL	(SP),STATUS	; ...and save it as the exit status
	04	DD	0386	1220		PUSHL	#4	; Push the partial arg count...
08E5		31	0388	1221		BRW	ERROR_EXIT	; ...and split this scene



```

0388 1223 :+
0388 1224 : The following code dynamically allocates enough memory for a unit block,
0388 1225 : a device dependent parameter area and I/O buffers. The unit block is inserted
0388 1226 : into the queue header UNIT LIST. It then initializes the unit block.
0388 1227 : A comment indicates where the device dependent parameters should be
0388 1228 : initialized. The unit block format is as follows:
0388 1229 :
0388 1230 :
0388 1231 : UETUNT$L_FLINK +-----+
0388 1232 : UETUNT$B_LINK +-----+
0388 1233 : UETUNT$B_TYPE +-----+
0388 1234 : UETUNT$W_SIZE +-----+ contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0388 1235 : UETUNT$B_FLAGS +-----+
0388 1236 : UETUNT$W_CHAN +-----+
0388 1237 : UETUNT$W_FUNC +-----+
0388 1238 : UETUNT$L_ITER +-----+ UETUNT$C_SIZE
0388 1239 : UETUNT$T_FILSPC +-----+
0388 1240 : UETUNT$K_FAB +-----+
0388 1241 : UETUNT$K_RAB +-----+
0388 1242 : UETUNT$K_DEVDEP +-----+
0388 1243 : UETUNT$K_READWRITE +-----+
0388 1244 : UETUNT$K_READWRITE +-----+
0388 1245 : UETUNT$K_READWRITE +-----+
0388 1246 : UETUNT$K_READWRITE +-----+
0388 1247 : UETUNT$K_READWRITE +-----+
0388 1248 : UETUNT$K_READWRITE +-----+
0388 1249 : UETUNT$K_READWRITE +-----+
0388 1250 : UETUNT$K_READWRITE +-----+
0388 1251 : UETUNT$K_READWRITE +-----+
0388 1252 : UETUNT$K_READWRITE +-----+
0388 1253 : UETUNT$K_READWRITE +-----+
0388 1254 : UETUNT$K_READWRITE +-----+
0388 1255 : UETUNT$K_READWRITE +-----+
0388 1256 : UETUNT$K_READWRITE +-----+
0388 1257 : UETUNT$K_READWRITE +-----+
0388 1258 : UETUNT$K_READWRITE +-----+
0388 1259 : UETUNT$K_READWRITE +-----+
0388 1260 : UETUNT$K_READWRITE +-----+
0388 1261 : UETUNT$K_READWRITE +-----+
0388 1262 : UETUNT$K_READWRITE +-----+
0388 1263 : UETUNT$K_READWRITE +-----+
0388 1264 : UETUNT$K_READWRITE +-----+
0388 1265 : UETUNT$K_READWRITE +-----+
0388 1266 : UETUNT$K_READWRITE +-----+
0388 1267 : UETUNT$K_READWRITE +-----+
0388 1268 : UETUNT$K_READWRITE +-----+
0388 1269 : UETUNT$K_READWRITE +-----+
0388 1270 : UETUNT$K_READWRITE +-----+
0388 1271 : UETUNT$K_READWRITE +-----+
0388 1272 : UETUNT$K_READWRITE +-----+
0388 1273 : UETUNT$K_READWRITE +-----+
0388 1274 : UETUNT$K_READWRITE +-----+
0388 1275 : UETUNT$K_READWRITE +-----+
0388 1276 : UETUNT$K_READWRITE +-----+
0388 1277 : UETUNT$K_READWRITE +-----+

```



```
038B 1279 60$:  
038B 1280  
038B 1281  
0198'CF 01A0'DF 5D 039C 1282  
56 01A0'CF D0 03A3 1283  
08 A6 01 90 03A8 1284  
01A4 8F B0 03AC 1285  
09 A6 03B0 1286  
14 A6 00A0'CF 90 03B2 1287  
00A4'DF 00A0'CF 28 03B8 1288  
15 A6 03BF 1289  
0094 8F 28 03C1 1290  
0110 C6 173C'CF 03C5 1291  
57 0110 C6 DE 03CB 1292  
58 0160 C6 DE 03D0 1293  
3C A8 57 D0 03D5 1294  
14 A6 90 03D9 1295  
34 A7 03DC 1296  
15 A6 DE 03DE 1297  
2C A7 03E1 1298  
03E3 1299  
03E3 1300 : Set the device dependent parameters in here  
03E3 1301 :  
FE93 31 03E3 1302  
$EXPREG_S PAGCNT = #PAGES,- ; Get a new node of demand zero memory  
RETADR = NEW_NODE  
INSQTI @NEW_NODE,UNIT_LIST ; Put the new node in the unit list  
MOVL NEW_NODE,R6 ; Save a copy of its address  
MOVB #1,DETUNT$B TYPE(R6) ; Set the structure type  
MOVW #UETUNT$C INDSIZ+DEVDEP_SIZE,- ; Set the structure size  
UETUNT$W SIZE(R6)  
MOVB DEVDSCL,UETUNT$T FILSPC(R6) ; Set the device name size  
MOVC3 DEVDSCL,@DEVDSCL+4,-  
UETUNT$T FILSPC+1(R6) ; Save the device name  
MOVC3 #FAB$C BLN+RAB$C BLN,-  
DUMMY FAB,UETUNT$C FAB(R6) ; Save a FAB and a RAB away  
MOVAL UETUNT$K_FAB(R6),R7 ; Save the FAB address  
MOVAL UETUNT$K_RAB(R6),R8 ; Save the RAB address  
MOVL R7,RAB$K_FAB(R8) ; Set the FAB address in the RAB  
MOVB UETUNT$T_FILSPC(R6),-  
FAB$B FNS(R7) ; Set the FNS field in the FAB  
MOVAL UETUNT$T_FILSPC+1(R6),-  
FAB$K_FNA(R7) ; Set the FNA field in the FAB  
BRW FOUND_IT ; Do the next UCB
```



```
03E6 1304 :  
03E6 1305 : Arrive here when we have the device configuration. In normal or loop forever  
03E6 1306 : mode, set a timer far enough in the future such that we can do a reasonable  
03E6 1307 : set of tests before the timer expires, but if our device gets hung, the  
03E6 1308 : program won't waste too much time before noticing. Let one-shot mode be a  
03E6 1309 : special case.  
03E6 1310 :  
03E6 1311 ALL_SET:  
0198'CF D5 03E6 1312 TSTL UNIT_LIST ; Anything to test?  
16 12 03EA 1313 BNEQ 10$ ; BR if yes  
033C'CF DF 03EC 1314 PUSHAL NOUNIT_SELECTED ; Else set up the error message...  
01 DD 03F0 1315 PUSHL #1 ; ...argument count...  
00741132 8F DD 03F2 1316 PUSHL #UETPS_TEXT!STSSK_ERROR ; ...signal name...  
03 DD 03F8 1317 PUSHL #3 ; ...and parameter count  
014E'CF 14 D0 03FA 1318 MOVL #SS$ BADPARAM,STATUS ; Set return status  
086E 31 03FF 1319 BRW ERROR_EXIT ; ...and give up, complaining  
000A'CF 04 A8 0402 1320 10$:  
0402 1321 BISW2 #SAFE TO_UPDM,FLAG ; OK safe to update UETINIDEV.DAT now  
0407 1322 $ASSIGN_S CHAN = CHAN,-  
0407 1323 DEVNAM = DEVDESC ; Get a channel  
0418 1324 $LKWSET_S INADR = CMDBLKDES ; Lock command block into WS  
0427 1325 ; 11/780 ucode bug in QUEUE  
0427 1326 ; Instructions  
52 00000000'GF 9A 0427 1327 MOVZBL G^IOC$GW_XFMXRATE,R2 ; Get the current max xfer rate  
12C7'CF 52 90 042E 1328 MOV B R2,W^CMDTBL+XFSB_CMT_RATE ; Set the data transfer rate  
53 00000100 8F 52 C3 0433 1329 SUBL3 R2,#256,R3 ; 256-max xfer rate into R3  
54 53 6E 043B 1330 CVTLD R3,R4 ; Convert to double float format  
56 32 50 043E 1331 MOVF #^F40.,R6 ; Set the double constant  
57 03C9'CF DE 0441 1332 MOVAL DR780,R7 ; Set default device name  
00DB'CF 00'8F 91 0446 1333 CMPB #DT$_DR750,DIBBUF+DIB$B_DEVTYPE ; Is it a 750?  
OC 12 044C 1334 BNEQ 15$ ; BR if not  
56 00004248 8F 50 044E 1335 MOVF #^F12.5,R6 ; Set new conversion rate  
57 03CD'CF DE 0455 1336 MOVAL DR750,R7 ; Set correct device name  
045A 1337 15$:  
0217'CF 56 54 67 045A 1338 DIVD3 R4,R6,W^RATE_FLOAT ; 40/(256-max xfer rate)  
03 DD 0460 1339 PUSHL #3 ; Push # of digits in the fraction  
0208'CF DF 0462 1340 PUSHAL W^RATE_BUF ; Push string storage desc adr  
0217'CF 7F 0466 1341 PUSHAQ W^RATE_FLOAT ; Push adr of floating number  
00000000'GF 03 FB 046A 1342 CALLS #3,G^FOR$CNV_OUT_F ; Make the number a string  
0471 1343 $FAO_S W^CS2,W^RATE_DESC,W^FAO_BUF,-  
0471 1344 R7,R2,#RATE_BUF ; Make up the message  
021F'CF DF 048E 1345 PUSHAL W^RATE_DESC ; Push the string address  
01 DD 0492 1346 PUSHL #1 ; Push the arg count  
00741133 8F DD 0494 1347 PUSHL #UETPS_TEXT!STSSK_INFO ; Push the signal name  
00000000'GF 03 FB 049A 1348 CALLS #3,G^LIB$SIGNAL ; Report the message  
04A1 1349 $CREMBX_S CHAN = MBCHAN ; make a mailbox  
04B4 1350 $GETCHN_S CHAN = MBCHAN,-  
04B4 1351 PRIBUF = DIB ; get the unit number of the mailbox  
04CA 1352 $CREPRC_S IMAGE = PROCESS,-  
04CA 1353 MBXUNT = DIBBUF+DIB$W_UNIT,-  
04CA 1354 ERROR = XFLDR_SYSS$ERROR_DESC,-  
04CA 1355 PIDADR = PID ; Load the ucode  
04F4 1356 $WAKE_S PIDADR = PID ; wake XFLOADER.EXE from the HIBER  
0501 1357 $SETIMR_S DAYTIM = ONEMIN,- ; Catch hungs by ucode loader process  
0501 1358 ASTADR = 100$,-  
0501 1359 REQIDT = PID  
0516 1360 $QIOW_S FUNC = #IOS_READVBLK,-
```



			0516	1361	EFN	= #1,-	
			0516	1362	CHAN	= MBCHAN,-	
			0516	1363	P1	= BUFFER,-	
			0516	1364	P2	= #256	
			0538	1365	SCANTIM	S REQIDT = PID	; read the ucode load results
15	0020'CF	E8	0548	1366	BLBS	BUFFER+ACCSL_FINALSTS,20\$	; Ucode loader process finished
	0020'CF	DD	054D	1367	PUSHL	BUFFER+ACCSL_FINALSTS	; check the load status
	0271'CF	DF	0551	1368	PUSHAL	ULOAD_FAILED	; Push the failure code
	01	DD	0555	1369	PUSHL	#1	; Push failure message address
00741132	8F	DD	0557	1370	PUSHL	#UETPS_TEXT!STSSK_ERROR	; Push arg count
	04	DD	055D	1371	PUSHL	#4	; Push the signal name
	070E	31	055F	1372	BRW	ERROR_EXIT	; Push the temp arg count
			0562	1373			; And die
			0562	1374	20\$:		
50	00000000'8F	D1	056D	1375	SOPEN	FAB = XFLDR SYSSERROR_FAB	; Did ucode loader find any errors?
	03	12	0574	1376	CMPL	#RMS\$_FNF,R0	; If there was no SYSSERROR file...
	0095	31	0576	1377	BNEQ	30\$	
			0579	1378	BRW	80\$	; ...then BR around file copy
	09 50	E8	0579	1379	30\$:		
	17D0'CF	DF	057C	1380	BLBS	R0,40\$	; BR if we can read the file
OBCA'CF	01	FB	0580	1381	PUSHAL	XFLDR SYSSERROR_FAB	; If we can't read it...
			0585	1382	CALLS	#1,RMS_ERROR	; ...then complain and bail out
			0585	1383	40\$:		
			0585	1384			
			0594	1385	\$CONNECT	RAB = XFLDR SYSSERROR_RAB,-	
			05A5	1386	ERR	= RMS_ERROR	
			05A5	1387	\$PUTMSG_S	MSGVEC = XFLDR_COPY_START	; Announce our intent to copy
50	00000000'8F	D1	05B0	1388	50\$:		
	26	13	05B7	1389	\$GET	RAB = XFLDR SYSSERROR_RAB	; Read a line from the file
	09 50	E8	05B9	1390	CMPL	#RMS\$_EOF,R0	; Are we beyond the file's end?
	1820'CF	DF	05BC	1391	BEQL	70\$	; BR out of loop if we are
OBCA'CF	01	FB	05C0	1392	BLBS	R0,60\$	; BR if we were able to read a record
			05C5	1393	PUSHAL	XFLDR SYSSERROR_RAB	; If we were not able to read it...
	1842'CF	B0	05C5	1394	CALLS	#1,RMS_ERROR	; ...then complain and bail out
	0014'CF		05C9	1395	60\$:		
			05CC	1396	MOVW	XFLDR_SYSSERROR_RAB+-	; Get the size of this line
			05DD	1397	RAB\$W-RSZ,BUFFER_PTR		
			05DF	1398	\$PUTMSG_S	MSGVEC = XFLDR_COPY_LINE	; Report contents of the line
			05DF	1399	BRB	50\$	; Loop for the next line
			05F0	1400	70\$:		
			05F0	1401	\$PUTMSG_S	MSGVEC = XFLDR_COPY_FINISH	; Announce the end of the file
			05FF	1402	\$CLOSE	FAB = XFLDR SYSSERROR_FAB,-	
			05FF	1403	ERR	= RMS_ERROR	
			060E	1404	\$ERASE	FAB = XFLDR SYSSERROR_FAB,-	
			060E	1405	ERR	= RMS_ERROR	
			060E	1406	80\$:		
			060E	1407	STRNLOG_S	LOGNAM = MODE,-	; Get the run mode
			0627	1408	RSLEN	= BUFFER_PTR,-	
001C'CF	20	8A	0627	1408	RSLBUF	= FAO BUF	
001C'CF	4F	8F	062C	1409	BICB2	#LC BITM,BUFFER	; Convert to upper case
	20	12	0632	1410	CMPL	#A70/,BUFFER	; Is this a one shot?
000A'CF	02	A8	0634	1411	BNEQ	TIME IT	; BR if not
	2C	11	0639	1412	BISW2	#TEST OVERM,FLAG	; End after one iteration
			063B	1413	BRB	RESTART	; Skip the SETIMR
			063B	1414	100\$:		
			063B	1415			
			063B	1416			
04 AC	0000	D5	063D	1417	; Reached by timer AST if microcode loader subprocess fails to		
					; return on schedule. Assume it's hung.		
					WORD	^M<>	
					TSTL	04(AP)	; Was the process even started?



```

01 12 0640 1418 BNEQ 110$ ; BR if it was
04 04 0642 1419 RET ; It wasn't - let mainline code handle
0643 1420 110$:
0077'CF DF 0643 1421 PUSHAL XFLDR_HUNG ; Set up an error message...
01 DD 0647 1422 PUSHL #1
00741132 8F DD 0649 1423 PUSHL #UETPS_TEXT!STSSK_ERROR
03 DD 064F 1424 PUSHL #3
061C 31 0651 1425 BRW ERROR_EXIT ; ...and bail out
0654 1426
0654 1427 TIME_IT:
0654 1428 $SETIMR_S DAYTIM = THREEMIN,- ; Set timer AST to 3 minutes
0654 1429 ASTADR = TEST_END,-
0654 1430 EFN = #EFN2

```



```
0667 1432 .SBTTL Test the DR780/DR750
0667 1433 RESTART:
0667 1434 :*****
0667 1435 :
0667 1436 : Device test specific code goes here.
0667 1437 :
0667 1438 : At this point the device designation is in location DEV_NAME pointed to by
0667 1439 : descriptor DEVDESC. The device is known to be supported and testable by this test.
0667 1440 : To leave successfully BRW SUC_EXIT, to leave in error BRW ERROR_EXIT.
0667 1441 :
0667 1442 :*****
0667 1443
000A'CF 10 8A 0667 1444 BICB2 #FPAC FLGM,FLAG ; clear the first packet AST flag
0667 1445 $SETIMR_S DAYTIM = TENSEC,-
0667 1446 EFN = #EFN2,-
0667 1447 ASTADR = HUNG_TEST,-
0667 1448 REQIDT = #1 ; set 10 sec watch dog timer in case
067F 1449 ; the clock jumpers are missing
067F 1450 $QIO_S FUNC = #IOS_STARTDATA,- ; Initialize the mapping pointers
067F 1451 CHAN = CHAN,- ; channel must have been assigned.
067F 1452 EFN = #EFN1,- ; EF
067F 1453 IOSB = DRIOSAT,- ; IO status block.
067F 1454 ASTADR = IO_COMPLETE,- ; ast to be taken when dr halts
067F 1455 P1 = CMDTBL,- ; address of the command table
067F 1456 P2 = #XFSK_CMT_LENGTH ; length of command table.
000A'CF 09 93 06A6 1457 BITB #CC_FLGM!ERR_FLGM,FLAG ; control C or error occurred?
03 13 06AB 1458 BEQL 5$ ; br if no
061D 31 06AD 1459 BRW ERROR_EXIT1 ; br if yes
06B0 1460 5$:
5A 056C'CF DE 06B0 1461 MOVAL PKT_TBL,R10 ; set pkt address pointer
5B 12D8'CF DE 06B5 1462 MOVAL INPTQH,R11 ; set the queue header pointer
59 6A D0 06BA 1463 MOVL (R10),R9 ; get the packet address
1C A9 D4 06BD 1464 CLRL XFSL_PKT_DSL(R9) ; init the DSL
06C0 1465 QRETRY ERROR = BADQUEUE,-
06C0 1466 INSQTI @ (R10)+,FREEQH ; put a free packet on the free queue
06D0 1467 .REPT PKT_COUNT-1
06D0 1468 MOVL (R10),R9 ; get the packet address
06D0 1469 CLRL XFSL_PKT_DSL(R9) ; init the DSL
06D0 1470 QRETRY ERROR = BADQUEUE,-
06D0 1471 INSQTI @ (R10)+,(R11) ; put a command packet on the input
59 6A D0 06D0 1472 .ENDR
12CF'DF 01 D0 07C0 1473 MOVL #1,@GOBIT ; give it heck!
07C5 1474 $WAITFR_S EFN = #EFN1 ; wait for further AST's or iteratio
000A'CF 09 93 07CE 1475 BITB #CC_FLGM!ERR_FLGM,FLAG ; control C or error occurred?
03 13 07D3 1476 BEQL 10$ ; br if no
04F5 31 07D5 1477 BRW ERROR_EXIT1 ; br if yes
07D8 1478 10$:
01AC'CF 0176'CF D6 07D8 1479 INCL ITERATION ; increment iteration count
00006030 8F D0 07DC 1480 MOVL #UNUSED_FUNC,PACK_REMOVED ; mask out unused packet types
0A46'CF 00 FB 07E5 1481 CALLS #0,PKT_CHECK ; check the packet status and count
000A'CF 08 93 07EA 1482 BITB #ERR_FLGM,FLAG ; were any DSL's bad?
03 13 07EF 1483 BEQL 20$ ; br if not
04D9 31 07F1 1484 BRW ERROR_EXIT1 ; fatal error
07F4 1485 20$:
01A8'CF 0D 91 07F4 1486 CMPB #PKT_COUNT,PKT_CNT ; right number of packets?
69 13 07F9 1487 BEQL 50$ ; br if yes
55 D4 07FB 1488 CLRL R5 ; set starting position
```



```

      58 11 DO 07FD 1489      MOVL #NO OF POS_PKTS,R8      ; set starting size
      52 01B0'CF DE 0800 1490      MOVAL ARG5,R2      ; set arg pointer
      82 82 0D DO 0805 1491      MOVL #PKT_COUNT,(R2)+      ; set expected pkt count
      82 0D 01A8'CF DO 0808 1492      MOVL PKT_CNT,(R2)+      ; set received pkt count
      0D 01A8'CF C3 080D 1493      SUBL3 PKT_CNT,#PKT_COUNT,(R2)+      ; set the argument count
      56 01AC'CF 58 55 EB 0813 1494 30$:      FFC R5,R8,PACK_REMOVED,R6      ; find a missing packet
      17 13 081A 1495      BEQL 40$
      58 11 56 C3 081C 1496      SUBL3 R6,#NO OF POS_PKTS,R8      ; update the size
      55 01 56 C1 0820 1497      ADDL3 R6,#1,R5      ; update the starting position
      56 08 C4 0824 1498      MULL2 #8,R6      ; make it a byte displacement
      000004E4'8F CO 0827 1500      ADDL2 #NAME_TBL,R6      ; make it an address
      82 66 DO 082E 1501      MOVL (R6),(R2)+      ; save it in the argument list
      E0 11 0831 1502      BRB 30$      ; get the next packet that is missin
      0833 1503 40$:
      0833 1504
      0833 1505
      0833 1506
      0833 1507      $FAOL_S CTRSTR = CS4,-
      007410E2 8F DO 084A 1508      MOVL #UETPS_ABENDD!STSSK_ERROR,-      ; convert into a string
      014E'CF 0850 1509      STATUS      ; set an exit status
      0200'CF DF 0853 1510      PUSHAL BADRPKT      ; push the constructed message
      01 DD 0857 1511      PUSHL #1      ; push arg count
      00741132 8F DD 0859 1512      PUSHL #UETPS_TEXT!STSSK_ERROR      ; push the signal name
      03 DD 085F 1513      PUSHL #3      ; push temp arg count
      040C 31 0861 1514      BRW ERROR_EXIT      ; bail out
      0864 1515 50$:
      1591'CF 1591'CF 92 0864 1516      MCOMB DIAG_CNTRL_MESS,DIAG_CNTRL_MESS      ; toggle the control message
      000A'CF 02 93 086B 1517      BITB #TEST_OVERM,FLAG      ; is the test over?
      03 13 0870 1518      BEQL 60$      ; br if no
      0115 31 0872 1519      BRW SUC_EXIT      ; br if yes
      FDEF 31 0875 1520 60$:
      0875 1521      BRW RESTART      ; do it again!
      0878 1522
      0878 1523 : Packet AST routine
      0878 1524
      0878 1525 PKT1_AST:
      54 000A'CF 04 OFFC 0878 1526      .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      1233'CF 0233'CF 007C 8F E2 087A 1527      BBSS #FPAC_FLGV,FLAG,10$      ; only check data on the first packe
      48 13 088A 1528      CMPC3 #124,OUTPUT_BUF+4,INPUT1_BUF+4      ; check the data
      014A'CF D6 088C 1529      BEQL 10$      ; br if OK
      7E 63 9A 0890 1530      INCL ERROR_COUNT      ; bump the cum. error cnt
      7E 61 9A 0893 1531      MOVZBL (R3),-(SP)      ; get bad byte
      7E 50 C3 0896 1532      MOVZBL (R1),-(SP)      ; get good byte
      0000007C 8F 50 C3 0896 1533      SUBL3 R0,#124,-(SP)      ; get the byte number
      00 DD 089E 1534      PUSHL #0      ; push the unit number
      00A0'CF DF 08A0 1535      PUSHAL DEVDSK      ; push the controller name
      000F0005 8F DD 08A4 1536      PUSHL #^XF0005      ; push arg count
      00748012 8F DD 08AA 1537      PUSHL #UETPS_DATAER!STSSK_ERROR      ; push the signal name
      014A'CF DD 08B0 1538      PUSHL ERROR_COUNT      ; push cumulative error count
      00A8'CF DF 08B4 1539      PUSHAL PROCESS_NAME
      00010002 8F DD 08B8 1540      PUSHL #^X10002      ; push arg count
      00748022 8F DD 08BE 1541      PUSHL #UETPS_ERBOXPROC!STSSK_ERROR      ; push the signal name
      00000000'GF 0B FB 08C4 1542      CALLS #11,G^CIB$SIGNAL      ; output the message
      00748012 8F DO 08CB 1543      MOVL #UETPS_DATAER!STSSK_ERROR,-
      014E'CF 08D1 1544      STATUS
      08D4 1545 10$:

```



```
007C 8F 00 122F'CF 00 2C 08D4 1546      MOVCS  #0,INPUT1_BUF,#0,#124,INPUT1_BUF+4 ; clear the data buffer
      1233'CF      04 08DD 1547      RET
      08E0 1548      ;
      08E1 1549      ; Watch dog timer will come to here if the DR does not complete one
      08E1 1550      ; execution of all packets within 10 seconds.
      08E1 1551      ;
      08E1 1552      HUNG_TEST:
      08E1 1553      .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; entry mask
      08E3 1554      BISB2  #ERR_FLGM,FLAG ; set the error flag
      08E8 1555      MOVL   #SSS_TIMEOUT,STATUS ; Set the exit status
      08F1 1556      INCL   ERROR_COUNT ; bump the error counter
      08F5 1557      PUSHAL TEST_HUNG ; push the constructed message
      08F9 1558      PUSHL  #^XF0001 ; push arg count
      08FF 1559      PUSHL  #UETPS_TEXT!STSSK_ERROR ; push the signal name
      0905 1560      PUSHL  ERROR_COUNT ; push cumulative error count
      0909 1561      PUSHAL PROCESS_NAME
      090D 1562      PUSHL  #^X10002 ; push the arg count
      0913 1563      PUSHL  #UETPS_ERBOXPROC!STSSK_ERROR ; push the signal name
      0919 1564      CALLS  #7,G^LIB$SIGNAL ; output the message
      0920 1565      $SETEF_S EFN=#EFN1 ; time to wake up
      0929 1566      RET ; go and fail
      092A 1567      IO_COMPLETE:
      092A 1568      .WORD  0 ; QIO completion ast entry point
      092C 1569      $CANTIM_S REQIDT=#1 ; stop the watch dog timer only
      0937 1570      CMPW   DRIOSTAT,#SSS_NORMAL ; check the IO status
      093C 1571      BEQL   10$ ; br if OK
      093E 1572      MOVW   DRIOSTAT,STATUS ; save the status
      0945 1573      INCL   ERROR_COUNT ; bump the error count
      0949 1574      PUSHL  DRIOSTAT ; push the DSL
      094D 1575      PUSHAL START_DATA_FAILED
      0951 1576      PUSHL  #^XF0001
      0957 1577      PUSHL  #UETPS_TEXT!STSSK_ERROR
      095D 1578      PUSHL  ERROR_COUNT ; push cumulative error count
      0961 1579      PUSHAL PROCESS_NAME
      0965 1580      PUSHL  #^X10002 ; push arg count
      096B 1581      PUSHL  #UETPS_ERBOXPROC!STSSK_ERROR ; push the signal name
      0971 1582      CALLS  #8,G^LIB$SIGNAL ; output the message
      0978 1583      BISB2  #ERR_FLGM,FLAG ; set error flag
      097D 1584      10$:
      097D 1585      CLRL   DRIOSTAT ; reset the DR's IO status block
      0981 1586      RET ; and return
      0982 1587      TEST_END:
      0982 1588      .WORD  0 ; entry mask
      0984 1589      BISB2  #TEST_OVERM,FLAG ; set the test ended flag
      0989 1590      RET ; return
      098A 1591      ;
```



```
001C'CF 20 8A 09A3 1598 BICB2 #LC_BITM,BUFFER ; Get the run mode
001C'CF 4C 8F 91 09A8 1599 CMPB #^A7L/,BUFFER ; Convert to upper case
                                40 12 09AE 1600 BNEQ 10$ ; Is this a loop for ever?
000A'CF 02 AA 09B0 1601 BICW2 #TEST_OVERM,FLAG ; BR if not
017A'CF D6 09B5 1602 INCL PASS ; Reset the termination flag
                                09B9 1603 $FAO_S CTRSTR = PASS_MSG,- ; Bump the pass count
                                09B9 1604 OUTLEN = BUFFER_PTR,-
                                09B9 1605 OUTBUF = FAO_BUF,-
                                09B9 1606 P1 = PASS,-
                                09B9 1607 P2 = ITERATION,-
                                09B9 1608 P3 = #0 ; Make the end of pass message
                                09D6 1609 PUSHAL BUFFER_PTR ; Push the string desc.
0014'CF DF 09DA 1610 PUSHL #1 ; Push arg count
                                00741133 8F DD 09DC 1611 PUSHL #UETPS_TEXT!STSSK_INFO ; Push the signal name
00000000'GF 03 FB 09E2 1612 CALLS #3,G^LIB$SIGNAL ; Print the end of pass message
0176'CF D4 09E9 1613 CLRL ITERATION ; Reset the iteration count
FC64 31 09ED 1614 BRW TIME_IT ; Do the next pass
56 0198'CF 00000198'8F C1 09F0 1615 10$: ADDL3 #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                                02 88 09FA 1617 BISB2 #UETUNT$M_TESTABLE,-
                                0B A6 09FC 1618 UETUNT$B_FLAGS(R6) ; Set the testable bit
014E'CF 10000001 8F D0 09FE 1619 MOVL #SS$ NORMAL!STSSM_INHIB_MSG,STATUS ; Set successful exit status
OA07 1620 $EXIT_S STATUS ; Exit with the status
```



```

0A12 1622 .SBTTL RANBUF
0A12 1623 :++
0A12 1624 : FUNCTIONAL DESCRIPTION:
0A12 1625 : This routine fills buffer TEST_DATA with random numbers.
0A12 1626 :
0A12 1627 : CALLING SEQUENCE:
0A12 1628 : BSBW RANBUF
0A12 1629 :
0A12 1630 : INPUT PARAMETERS:
0A12 1631 : NONE
0A12 1632 :
0A12 1633 : OUTPUT PARAMETERS:
0A12 1634 : BUFSIZ bytes of random data are left in buffer TEST_DATA
0A12 1635 :
0A12 1636 :--
0A12 1637
0A12 1638 RANBUF:
0A12 1639 MOVAL TEST_DATA,R2 ; set buffer adr
0A17 1640 MOVZWL #BUFSIZ/4,R3 ; BUFSIZ/4 bytes is the size
0A1C 1641 10$:
0A1C 1642 ADDL2 RANDOM2,RANDOM1 ; make a new random number
0A23 1643 MOVL RANDOM1,(R2)+ ; put it in the buffer
0A28 1644 SOBGTR R3,10$ ; do the whole thing!
0A2B 1645 RSB ; return

```



```

0A2C 1647 .SBTTL BADQUEUE
0A2C 1648 :++
0A2C 1649 : FUNCTIONAL DESCRIPTION:
0A2C 1650 : This routine indicates a bad queue entry has been discovered by the
0A2C 1651 : QRETRY macro and an error is reported.
0A2C 1652 :
0A2C 1653 : CALLING SEQUENCE:
0A2C 1654 : BRW BADQUEUE
0A2C 1655 :
0A2C 1656 : INPUT PARAMETERS:
0A2C 1657 : ERROR_COUNT = current cumulative error count
0A2C 1658 :
0A2C 1659 : OUTPUT PARAMETERS:
0A2C 1660 : ERROR_COUNT = bumped by one
0A2C 1661 :
0A2C 1662 :--
0A2C 1663
0A2C 1664 BADQUEUE:
007410E2 8F D0 0A2C 1665 MOVL #UETPS_ABENDD!ST$K_ERROR,-
014E'CF 0A32 1666 STATUS ; set the status code
0472'CF DF 0A35 1667 PUSHAL BADQUE ; push message address
01 DD 0A39 1668 PUSHL #1 ; push arg count
00741132 8F DD 0A3B 1669 PUSHL #UETPS_TEXT!ST$K_ERROR ; push signal name
03 DD 0A41 1670 PUSHL #3 ; Push temp arg count
022A 31 0A43 1671 BRW ERROR_EXIT

```



```
0A46 1673 ;
0A46 1674 ;
0A46 1675 ;++ .SBTTL PKT_CHECK
0A46 1676 ; FUNCTIONAL DESCRIPTION:
0A46 1677 ; Routine to check DR packet status off the termination queue.
0A46 1678 ; Each packet is removed from the termination queue and the
0A46 1679 ; DSL is checked for XF$M_IOS_SUCCE$ and XF$M_IOS_CMDSTD. A
0A46 1680 ; total count is maintained for each call to the routine at
0A46 1681 ; location PKT_CNT.
0A46 1682 ;
0A46 1683 ; CALLING SEQUENCE:
0A46 1684 ; CALLS #0,PKT_CHECK
0A46 1685 ;
0A46 1686 ; INPUT PARAMETERS:
0A46 1687 ; TERMQH = termination queue head
0A46 1688 ;
0A46 1689 ; OUTPUT PARAMETERS:
0A46 1690 ; PKT_CNT = number of packets serviced for this call
0A46 1691 ; FLAG = BIT1 set if an error is encountered
0A46 1692 ; PACK_REMOVED = bit mask record of which packets were removed
0A46 1693 ; from the termination queue
0A46 1694 ;
0A46 1695 ;--
0A46 1696 PKT_CHECK:
0A46 1697 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
01A8'CF OFFC 0A48 1698 CLR B PKT_CNT ; set packet count to zero
0A4C 1699 PC1:
0A4C 1700 QRETRY ERROR = BADQUEUE,-
0A4C 1701 REMQHI TERMQH,R2 ; get a packet from the queue
01 1C 0A5C 1702 BVC 5$ ; br if an entry removed
04 0A5E 1703 RET ; return
0A5F 1704 5$:
01A8'CF 96 0A5F 1705 INCB PKT_CNT ; bump the packet count
1C A2 08 D3 0A63 1706 BITL #XF$M_PKT_FREQPK,XF$M_PKT_DSL(R2) ; is it a free queue packet?
05 13 0A67 1707 BEQL 10$ ; br if not
53 10 D0 0A69 1708 MOVL #16,R3 ; set the free packet index
06 11 0A6C 1709 BRB 15$ ; carry on
04 00 EF 0A6E 1710 10$:
53 0A A2 0A71 1711 EXTZV #XF$V_PKT_FUNC,#XF$S_PKT_FUNC,-
0A74 1712 XF$B_PKT_CMDCTL(R2),R3 ; get the packet command
01AC'CF 01 53 01 F0 0A74 1713 15$:
53 53 08 C4 0A7B 1714 INSV #1,R3,#1,PACK_REMOVED ; record it's removal
53 000004E4'8F C0 0A7E 1715 MULL2 #8,R3 ; make it an index
54 53 04 C1 0A85 1716 ADDL2 #NAME_TBL,R3 ; make it an ascic packet type pointer
1C A2 64 D1 0A89 1717 ADDL3 #4,R3,R4 ; make the status address
BD 13 0A8B 1718 CMPL (R4),XF$M_PKT_DSL(R2) ; is the DSL correct?
007410E2 8F D0 0A8D 1719 BEQL PC1 ; br if OK
014E'CF 0A95 1720 20$:
014A'CF D6 0A98 1721 MOVL #UETP$_ABENDD!ST$K_ERROR,-
0A9C 1722 STATUS ; set the status code
0A9C 1723 INCL ERROR_COUNT ; bump the error counter
0A9C 1724 $FAO_S
0A9C 1725 CTRSTR = CS,-
0A9C 1726 OUTLEN = BADRPKT,-
0A9C 1727 OUTBUF = FAO_BUF,-
0A9C 1728 P1 = (R3),-
0A9C 1729 P2 = XF$M_PKT_DSL(R2),-
P3 = (R4) ; create the error string
```



UETDR7800  
V04-000

L 16

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 42  
PKT\_CHECK 5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (14)

0200'CF	DF	0AB6	1730	PUSHAL	BADRPKT	; set message address
000F0001 8F	DD	0ABA	1731	PUSHL	#^XF0001	; set arg count
00741132 8F	DD	0ACO	1732	PUSHL	#UETPS_TEXT!STSSK_ERROR	; set signal name
014A'CF	DD	0AC6	1733	PUSHL	ERROR_COUNT	; push cumulative error count
00A8'CF	DF	0ACA	1734	PUSHAL	PROCESS_NAME	
00010002 8F	DD	0ACE	1735	PUSHL	#^X10002	; push arg count
00748022 8F	DD	0AD4	1736	PUSHL	#UETPS_ERBOXPROC!STSSK_ERROR	; set signal name
00000000'GF 07	FB	0ADA	1737	CALLS	#7,G^LIB\$SIGNAL	; output the message
000A'CF 08	88	0AE1	1738	BISB2	#ERR_FLGM,FLAG	; set the error flag
	04	0AE6	1739	RET		; return



```

OAE7 1741      .SBTTL System Service Exception Handler
OAE7 1742      :++
OAE7 1743      : FUNCTIONAL DESCRIPTION:
OAE7 1744      :   This routine is executed if a software or hardware exception occurs or
OAE7 1745      :   if a LIB$SIGNAL system service is used to output a message.
OAE7 1746      :
OAE7 1747      : CALLING SEQUENCE:
OAE7 1748      :   Entered via an exception from the system
OAE7 1749      :
OAE7 1750      : INPUT PARAMETERS:
OAE7 1751      :   ERROR_COUNT = previous cumulative error count
OAE7 1752      :
OAE7 1753      :   AP ---->
OAE7 1754      :
OAE7 1755      :   SIGNAL ARY PNT
OAE7 1756      :   MECH ARY PNT
OAE7 1757      :
OAE7 1758      :   4
OAE7 1759      :   ESTABLISH FP
OAE7 1760      :   DEPTH
OAE7 1761      :   R0
OAE7 1762      :   R1
OAE7 1763      :   N
OAE7 1764      :   CONDITION NAME
OAE7 1765      :   N-3 ADDITIONAL
OAE7 1766      :   LONG WORD ARGS
OAE7 1767      :   PC
OAE7 1768      :   PSL
OAE7 1769      :
OAE7 1770      :
OAE7 1771      :
OAE7 1772      :
OAE7 1773      :
OAE7 1774      :
OAE7 1775      :
OAE7 1776      :
OAE7 1777      :
OAE7 1778      :
OAE7 1779      :
OAE7 1780      : IMPLICIT INPUTS:
OAE7 1781      :   NONE
OAE7 1782      :
OAE7 1783      : OUTPUT PARAMETERS:
OAE7 1784      :   NONE
OAE7 1785      :
OAE7 1786      : IMPLICIT OUTPUTS:
OAE7 1787      :   NONE
OAE7 1788      :
OAE7 1789      : COMPLETION CODES:
OAE7 1790      :   SSS_NORMAL if it's a UETP condition or RMS error.
OAE7 1791      :   Error status from exception, otherwise.
OAE7 1792      :
OAE7 1793      : SIDE EFFECTS:
OAE7 1794      :   May branch to ERROR_EXIT.
OAE7 1795      :   May print a message.
OAE7 1796      :--
OAE7 1797

```

Mechanism Array

Signal Array



Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



00741130	8F	DD	OBA2	1855	PUSHL	#UETP\$ TEXT	; ...why the System Service failed
00	5A	FO	OBA8	1856	INSV	R10,#STSSV SEVERITY,-	; Give the message...
6E	03		OBAB	1857		#STSS\$ SEVERITY,(SP)	; ...the correct severity code
58	03	D0	OBAD	1858	MOVL	#3,R8	; Count the number of args we pushed
	05	11	OBBO	1859	BRB	70\$	
			OB82	1860			60\$:
	5A	DD	OB82	1861	PUSHL	R10	; Save SS failure code
58	01	D0	OB84	1862	MOVL	#1,R8	; Count the number of args we pushed
			OB87	1863			70\$:
57	66	04	C5	OB87	MULL3	#4,CHF\$L_SIG_ARGS(R6),R7	; Convert longwords to bytes
	5E	57	C2	OB8B	SUBL2	R7,SP	; Save the current signal array...
6E	04	A6	57	28	MOVCL	R7,CHF\$L_SIG_NAME(R6),(SP)	; ...on the stack
7E	66	58	C1	OB83	ADDL3	R8,CHF\$L_SIG_ARGS(R6),-(SP)	; Push the current arg count
	00A6	31	OB87	1868	BRW	ERROR_EXIT	



```

OBCA 1870      .SBTTL RMS Error Handler
OBCA 1871      :++
OBCA 1872      : FUNCTIONAL DESCRIPTION:
OBCA 1873      :   This routine handles error returns from RMS calls.
OBCA 1874      :
OBCA 1875      : CALLING SEQUENCE:
OBCA 1876      :   Called by RMS when a file processing error is found.
OBCA 1877      :
OBCA 1878      : INPUT PARAMETERS:
OBCA 1879      :   The FAB or RAB associated with the RMS call.
OBCA 1880      :
OBCA 1881      : IMPLICIT INPUTS:
OBCA 1882      :   NONE
OBCA 1883      :
OBCA 1884      : OUTPUT PARAMETERS:
OBCA 1885      :   NONE
OBCA 1886      :
OBCA 1887      : IMPLICIT OUTPUTS:
OBCA 1888      :   Error message
OBCA 1889      :
OBCA 1890      : COMPLETION CODES:
OBCA 1891      :   NONE
OBCA 1892      :
OBCA 1893      : SIDE EFFECTS:
OBCA 1894      :   Program may exit, depending on severity of the error.
OBCA 1895      :
OBCA 1896      :--
OBCA 1897
OBCA 1898      RMS_ERROR:
OBCA 1899      .WORD      ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OBCA 1900
OBCA 1901      MOVL      4(AP),R6 ; See whether we're dealing with...
OBCA 1902      CMPB      #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
OBCA 1903      BNEQ      10$ ; BR if it's a RAB
OBCA 1904      MOVAL     FILE,R7 ; FAB-specific code: text string...
OBCA 1905      MOVL      R6,R8 ; ...address of FAB...
OBCA 1906      PUSHL     FAB$STV(R6) ; ...STV field for error...
OBCA 1907      PUSHL     FAB$STS(R6) ; ...STS field for error...
OBCA 1908      MOVL      FAB$STS(R6),STATUS ; ...and save the error code
OBCA 1909      BRB       COMMON ; FAB and RAB share other code
OBCA 1910      10$:
OBCA 1911      MOVAL     RECORD,R7 ; RAB-specific code: text string...
OBCA 1912      MOVL      RAB$FAB(R6),R8 ; ...address of associated FAB...
OBCA 1913      PUSHL     RAB$STV(R6) ; ...STV field for error...
OBCA 1914      PUSHL     RAB$STS(R6) ; ...STS field for error...
OBCA 1915      MOVL      RAB$STS(R6),STATUS ; ...and save the error code
OBCA 1916      COMMON:
OBCA 1917      MOVZBL     FAB$B_FNS(R8),R10 ; Get the file name size
OBCA 1918      $FAO_S      CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
OBCA 1919      OUTLEN = BUFFER_PTR,-
OBCA 1920      OUTBUF = FAO_BUF,-
OBCA 1921      P1 = R7,-
OBCA 1922      P2 = R10,-
OBCA 1923      P3 = FAB$FNA(R8)
OBCA 1924      PUSHAL     BUFFER_PTR ; ...and arguments for ERROR_EXIT...
OBCA 1925      PUSHL      #1 ; ...
OBCA 1926      PUSHL      #UETP$_TEXT ; ...

56 04 AC DO OBCA 1901
66 03 91 OBD0 1902
16 12 OBD3 1903
57 041E'CF DE OBD5 1904
58 56 DO OBD4 1905
OC A6 DD OBD0 1906
08 A6 DD OBE0 1907
014E'CF 08 A6 DO OBE3 1908
15 11 OBE9 1909
OBE8 1910
57 042A'CF DE OBE8 1911
58 3C A6 DO OBF0 1912
OC A6 DD OBF4 1913
08 A6 DD OBF7 1914
014E'CF 08 A6 DO OBFA 1915
5A 34 A8 9A OC00 1916
OC04 1917
OC04 1918
OC04 1919
OC04 1920
OC04 1921
OC04 1922
OC04 1923
0014'CF DF OC1E 1924
01 DD OC22 1925
00741130 8F DD OC24 1926
```



UETDR7800  
V04-000

E 1

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 47  
RMS Error Handler 5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (16)

59	00	EF	0C2A	1927	EXTZV	#STSSV_SEVERITY,-	
	03		0C2C	1928		#STSSS_SEVERITY,-	
	014E'CF		0C2D	1929		STATUS,R9	; ...get the severity code...
	6E	59	88	0C31	BISB2	R9,(SP)	; ...and add it into the signal name
		05	DD	0C34	PUSHL	#5	; Current arg count
	0037	31	0C36	1932	BRW	ERROR_EXIT	



```

OC39 1934 .SBTTL CTRL/C Handler
OC39 1935 :++
OC39 1936 : FUNCTIONAL DESCRIPTION:
OC39 1937 : This routine handles CTRL/C AST's
OC39 1938 :
OC39 1939 : CALLING SEQUENCE:
OC39 1940 : Called via AST
OC39 1941 :
OC39 1942 : INPUT PARAMETERS:
OC39 1943 : NONE
OC39 1944 :
OC39 1945 : IMPLICIT INPUTS:
OC39 1946 : NONE
OC39 1947 :
OC39 1948 : OUTPUT PARAMETERS:
OC39 1949 : NONE
OC39 1950 :
OC39 1951 : IMPLICIT OUTPUTS:
OC39 1952 : NONE
OC39 1953 :
OC39 1954 : COMPLETION CODES:
OC39 1955 : NONE
OC39 1956 :
OC39 1957 : SIDE EFFECTS:
OC39 1958 : NONE
OC39 1959 :
OC39 1960 : --
OC39 1961 :
OC39 1962 CCASTHAND:
OFFC OC39 1963 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OC3B 1964
02B4'CF DF OC3B 1965 PUSHAL CNTRLMSG ; Set message pointer
01 DD OC3F 1966 PUSHL #1 ; Set arg count
00741130 8F DD OC41 1967 PUSHL #UETPS_TEXT!ST$K_WARNING ; Set signal name
00 DD OC47 1968 PUSHL #0 ; Indicate an abnormal termination
00A8'CF DF OC49 1969 PUSHAL PROCESS_NAME ; ...
02 DD OC4D 1970 PUSHL #2 ; ...
007410E0 8F DD OC4F 1971 PUSHL #UETPS_ABEND!ST$K_WARNING ; ...
00000000'GF 07 FB OC55 1972 CALLS #7,G^LIB$SIGNAL ; Output the message
DO OC5C 1973 MOVL #<ST$M_INHIB_MSG!- ; Set the exit status
OC5D 1974 SSS CONTROLC=-
OC5D 1975 ST$K_SUCCESS+ST$K_WARNING>,-
014E'CF 10000650 8F OC5D 1976 STATUS
OC65 1977 $EXIT_S STATUS ; Terminate program cleanly

```



```

OC70 1979      .SBTTL Error Exit
OC70 1980      :++
OC70 1981      : FUNCTIONAL DESCRIPTION:
OC70 1982      :   This routine prints an error message and exits.
OC70 1983      :
OC70 1984      : CALLING SEQUENCE:
OC70 1985      :   MOVx error status value,STATUS
OC70 1986      :   PUSHx error specific information on the stack
OC70 1987      :   PUSHL current argument count
OC70 1988      :   BRW ERROR_EXIT
OC70 1989      :
OC70 1990      : INPUT PARAMETERS:
OC70 1991      :   Arguments to LIB$SIGNAL, as above
OC70 1992      :
OC70 1993      : IMPLICIT INPUTS:
OC70 1994      :   NONE
OC70 1995      :
OC70 1996      : OUTPUT PARAMETERS:
OC70 1997      :   Message to SYS$OUTPUT and SYS$ERROR
OC70 1998      :
OC70 1999      : IMPLICIT OUTPUTS:
OC70 2000      :   Program exit
OC70 2001      :
OC70 2002      : COMPLETION CODES:
OC70 2003      :   NONE
OC70 2004      :
OC70 2005      : SIDE EFFECTS:
OC70 2006      :   NONE
OC70 2007      :
OC70 2008      :--
OC70 2009      :
OC70 2010      : ERROR_EXIT:
OC70 2011      :
OC70 2012      : $SETAST_S ENBFLG = #0 ; Disable AST's
15 000A'CF 05 E0 OC79 2013 BBS #BEGIN_MSGV,FLAG,10$ ; BR if "begin" msg already printed
OC70 2014 CLRL -(SP) ; Set the time stamp flag
OC70 2015 PUSHAL TEST_NAME ; Set the test name
OC70 2016 PUSHL #2 ; Push the argument count
OC70 2017 PUSHL #UETP$ BEGIN!ST$K_SUCCESS ; Set the message code
00000000'GF 04 FB OC8D 2018 CALLS #4,G^LIB$SIGNAL ; Print the startup message
OC70 2019 10$:
OC70 2020 ADDL3 (SP)+,#8,ARG_COUNT ; Get total # args, pop partial count
OC70 2021 INCL ERROR_COUNT ; Keep running error count
OC70 2022 PUSHL #0 ; Push the time parameter
OC70 2023 PUSHAL PROCESS_NAME ; Push test name...
OC70 2024 PUSHL #^XF0002 ; ...arg count...
OC70 2025 PUSHL #UETP$ ABEND!ST$K_ERROR ; ...and signal name
OC70 2026 PUSHL ERROR_COUNT ; Finish off arg list...
OC70 2027 PUSHAL PROCESS_NAME ; ...
OC70 2028 PUSHL #^X10002 ; ...
OC70 2029 PUSHL #UETP$ ERBOXPROC!ST$K_ERROR ; ...for error box message
00000000'GF 0192'CF FB OCC4 2030 CALLS ARG_COUNT,G^LIB$SIGNAL ; Truly bitch
OC70 2031 OCCD 2031
OC70 2032 ERROR_EXIT1:
OC70 2033 TSTL STATUS ; Did we exit with an error code?
OC70 2034 BNEQ 20$ ; BR if we did
007410E2 8F D0 OCD3 2035 MOVL #UETP$ _ABEND!ST$K_ERROR,- ; Supply a generic one otherwise

```



UETDR7800  
V04-000

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 H 1  
Error Exit 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 50  
5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (18)

014E'CF	10000000 8F	C8	OCD9 2036	STATUS
			OCDC 2037 20\$:	
			OCDC 2038	BISL #STS\$M-INHIB_MSG,STATUS ; Don't print messages twice!
			OCE5 2039	\$EXIT_S STATUS ; Exit in error



```
OCF0 2041 .SBTTL Exit Handler
OCF0 2042 :++
OCF0 2043 : FUNCTIONAL DESCRIPTION:
OCF0 2044 : This routine handles cleanup at exit. If the MODE logical name is
OCF0 2045 : equated to "ONE", the routine will update the test flag in the
OCF0 2046 : UETINIDEV.DAT file depending on the UETUNTSM_TESTABLE flag state in the
OCF0 2047 : UETUNT$B_FLAGS field of the unit block for each unit for the device
OCF0 2048 : under test.
OCF0 2049 :
OCF0 2050 : CALLING SEQUENCE:
OCF0 2051 : Invoked automatically by $EXIT System Service.
OCF0 2052 :
OCF0 2053 : INPUT PARAMETERS:
OCF0 2054 : STATUS contains the exit status.
OCF0 2055 : FLAG has synchronizing bits.
OCF0 2056 : DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
OCF0 2057 :
OCF0 2058 : IMPLICIT INPUTS:
OCF0 2059 : UNIT_LIST points to the head of a doubly linked circular list of unit
OCF0 2060 : blocks for the device under test.
OCF0 2061 :
OCF0 2062 : OUTPUT PARAMETERS:
OCF0 2063 : NONE
OCF0 2064 :
OCF0 2065 : IMPLICIT OUTPUTS:
OCF0 2066 : Various files are de-accessed and the process name is reset.
OCF0 2067 : If the MODE logical name is equated to "ONE", the routine will update
OCF0 2068 : the test flag in the UETINIDEV.DAT file depending on the
OCF0 2069 : UETUNTSM_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
OCF0 2070 : block for each unit for the device under test.
OCF0 2071 :
OCF0 2072 : COMPLETION CODES:
OCF0 2073 : NONE
OCF0 2074 :
OCF0 2075 : SIDE EFFECTS:
OCF0 2076 : NONE
OCF0 2077 :
OCF0 2078 : --
OCF0 2079 :
OCF0 2080 : EXIT_HANDLER:
OFFC OCF0 2081 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OCF2 2082
OCF2 2083 $SETSM_S ENBFLG = #0 ; Turn off System Service failure mode
OCFB 2084 $SETAST_S ENBFLG = #0 ; Disable AST's
OD04 2085 $STRNLOG_S LOGNAM = MCDE,- ; Get the run mode
OD04 2086 RSLLEN = BUFFER_PTR,-
OD04 2087 RSLBUF = FAO_BUF
001C'CF 20 8A OD1D 2088 BICB2 #LC_BITM,BUFFER ; Convert to upper case
001C'CF 4F 8F 91 OD22 2089 CMPB #^A70/,BUFFER ; Is this a one shot?
03 03 13 OD28 2090 BEQL 10$ ; BR if yes...
00B8 31 OD2A 2091 BRW END_UPDATE ; ...else don't update UETINIDEV.DAT
03 000A'CF 02 E0 OD2D 2092 10$:
00AF 31 OD2D 2093 BBS #SAFE_TO_UPDV,FLAG,20$ ; Only update if it's safe
5A 16A0'CF 02 E0 OD33 2094 BRW END_UPDATE ; Else forget it
1E AA 02 90 OD36 2095 20$:
OD36 2096 MOVAL INI RAB,R10 ; Set the RAB address
OD3B 2097 MOVB #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
```



Address	Hex	Asm	Comment
10 AA	16E4'CF	06 28 0D3F 2098	MOV C3 #6,DDB_RFA,RAB\$W_RFA(R10) ; Set RFA to DDB line
		0D46 2099	\$GET RAB = (R10) ; Go back to the DDB record
	75 50	E9 0D4F 2100	BLBC R0,UPDATE_FAILED ; If failure then forget it
	1E AA 00	90 0D52 2101	MOVB #RAB\$C_SEQ,RAB\$B_RAC(R10) ; Set back to sequential mode
5B 0198'CF	00000198'8F	C1 0D56 2102	ADDL3 #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
		59 D4 0D60 2103	CLRL R9 ; Init a counter
		0D62 2104	UNIT_LOOP:
	01	E1 0D62 2105	BBC #UETUNT\$V_TESTABLE,- ; BR if this unit is not testable
	02 0B AB	0D64 2106	UETUNT\$B_FLAGS(R11),10\$
	59	D6 0D67 2107	INCL R9 ; Count testable units
		0D69 2108	10\$:
	5B 6B	C0 0D69 2109	ADDL2 (R11),R11 ; Next unit block
00000198'8F	5B	D1 0D6C 2110	CMPL R11,#UNIT_LIST ; Are we full circle in the list?
	ED	12 0D73 2111	BNEQ UNIT_LOOP ; BR if not
	59	D5 0D75 2112	TSTL R9 ; Any testable units?
	12	12 0D77 2113	BNEQ 20\$ ; BR if yes...
0020'CF	4E 8F	90 0D79 2114	MOVB #^A/N/,BUFFER+4 ; ...else disable the DDB record...
		0D7F 2115	\$UPDATE RAB = (R10) ; ...here
	3C 50	E9 0D88 2116	BLBC R0,UPDATE_FAILED ; If error then forget it
		0D8B 2117	20\$:
	5B 6B	C0 0D8B 2118	ADDL2 (R11),R11 ; Next unit block
00000198'8F	5B	D1 0D8E 2119	CMPL R11,#UNIT_LIST ; Are we full circle in the list?
	4E	13 0D95 2120	BEQL END_UPDATE ; BR if yes
		0D97 2121	\$GET RAB = (R10) ; Get a record
	24 50	E9 0DA0 2122	BLBC R0,UPDATE_FAILED ; If error then forget it
001C'CF	20	8A 0DA3 2123	BICB2 #LC_BITM,BUFFER ; Convert to uppercase
001C'CF	55 8F	91 0DA8 2124	CMPB #^A7U/,BUFFER ; Is it a UCB record?
	35	12 0DAE 2125	BNEQ END_UPDATE ; BR if not
	01	E0 0DB0 2126	BBS #UETUNT\$V_TESTABLE,- ; BR if this unit is testable...
	D6 0B AB	0DB2 2127	UETUNT\$B_FLAGS(R11),20\$
0020'CF	4E 8F	90 0DB5 2128	MOVB #^A/N/,BUFFER+4 ; ...else disable the UCB record...
		0DBB 2129	\$UPDATE RAB = (R10) ; ...here
	C4 50	E8 0DC4 2130	BLBS R0,20\$ ; Look at the next record if no error
		0DC7 2131	UPDATE_FAILED:
	0C AA	DD 0DC7 2132	PUSHL RAB\$L_STV(R10) ; Do a simple message...
	50	DD 0DCA 2133	PUSHL R0 ; ...to tell of the failure
	03D1'CF	DF 0DCC 2134	PUSHAL INIDEV_UPDERR
	01	DD 0DD0 2135	PUSHL #1
	00	EF 0DD2 2136	EXTZV #STSS\$V_SEVERITY,- ; Copy the severity from RMS status...
	7E 50	03 0DD4 2137	#STSS\$SEVERITY,R0,-(SP)
6E 00741130	8F	C8 0DD7 2138	BISL2 #UETP\$TEXT,(SP) ; ...to our message
00000000'GF	05	FB 0DDE 2139	CALLS #5,G^LIB\$SIGNAL
		0DE5 2140	END_UPDATE:
	00	DD 0DE5 2141	PUSHL #0 ; Set the time flag
	000F'CF	DF 0DE7 2142	PUSHAL TEST_NAME ; Push the test name
	02	DD 0DEB 2143	PUSHL #2 ; Push arg count
	00	EF 0DED 2144	EXTZV #STSS\$V_SEVERITY,- ; Push the proper exit severity...
	03	0DEF 2145	#STSS\$SEVERITY,-
	7E 014E'CF	0DF0 2146	STATUS,-(SP)
6E 00741080	8F	C8 0DF4 2147	BISL2 #UETP\$_ENDED,(SP) ; ...and use it in our message code
	04	DD 0DFB 2148	PUSHL #4
	51 5E	D0 0DFD 2149	MOVL SP,R1
		0E00 2150	\$PUTMSG S MSGVEC = (R1) ; Output the message
		0E0F 2151	\$SETPRN S PRCNAM = ACNT_NAME ; Reset the process name
		0E1A 2152	RET ; That's all folks!
		0E1B 2153	
		0E1B 2154	.END UETDR7800



UETDR7800  
Symbol table

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 53  
5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (19)

\$\$\$.TAB = 00001820 R 03  
\$\$\$.TABEND = 00001864 R 03  
\$\$\$.TMP = 00000000  
\$\$\$.TMP1 = 00000001  
\$\$\$.TMP2 = 0000006A  
\$\$\$.TMPX = 00000016 R 04  
\$\$\$.TMPX1 = 0000000D  
\$\$T1 = 00000000  
\$\$T2 = 00000006  
ACCSL\_FINALSTS = 00000004  
ACNT\_NAME = 00000000 R 02  
ALL\_SET = 000003E6 R 05  
ARGS = 000001B0 R 03  
ARG\_COUNT = 00000192 R 03  
BADQUE = 00000472 R 02  
BADQUEUE = 00000A2C R 05  
BADRPKT = 00000200 R 03  
BEGIN\_MSGM = 00000020  
BEGIN\_MSGV = 00000005  
BUFBLE = 0000022F R 03  
BUFBLSIZ = 00001080  
BUFFER = 0000001C R 03  
BUFFER\_PTR = 00000014 R 03  
BUFSIZ = 00000800  
CCASTHAND = 00000C39 R 05  
CC\_FLGM = 00000001  
CC\_FLGV = 00000000  
CHAN = 00000004 R 03  
CHFSL\_SIGARGLST = 00000004  
CHFSL\_SIG\_ARG1 = 00000008  
CHFSL\_SIG\_ARGS = 00000000  
CHFSL\_SIG\_NAME = 00000004  
CLR\_RAND\_ENABLE = 0000065C R 02  
CLR\_SELF\_PKT = 00001350 R 03  
CLR\_SELF\_TEST = 000005F7 R 02  
CMDBLK = 000012D8 RG 03  
CMDBLKDES = 000004DC R 02  
CMDBLKEND = 000015BC R 03  
CMDBLSIZ = 000002E4  
CMDTBL = 000012AF R 03  
CMDTBLSIZ = 00000020  
CNTRLMSG = 000002B4 R 02  
COMMON = 00000C00 R 05  
CONTROLLER = 00000031 R 02  
CONT\_DESC = 00000416 R 02  
CS = 0000013F R 02  
CS1 = 000001A5 R 02  
CS1L = 000000C4  
CS2 = 000001B7 R 02  
CS3 = 000001FC R 02  
CS4 = 0000020B R 02  
DDB\_RFA = 000016E4 R 03  
DEAD\_CTRLNAME = 000002F5 R 02  
DEVSV\_TRM = 00000002  
DEVDEP\_SIZE = 00000000  
DEVVSC = 000000A0 R 03  
DEVNAM\_LEN = 0000016C R 03

DEV\_NAME = 000000BF R 03  
DIAG\_CNTRL\_MESS = 00001591 R 03  
DIAG\_READ\_INT = 0000060E R 02  
DIAG\_REA\_PKT = 00001390 R 03  
DIAG\_WRT\_INT = 0000061E R 02  
DIAG\_WRI\_PKT = 00001370 R 03  
DIAG\_WRT\_CNTRL = 00000639 R 02  
DIAG\_WRT\_PKT = 00001570 R 03  
DIB = 000000CE R 03  
DIBSB\_DEVCLASS = 00000004  
DIBSB\_DEVTYPE = 00000005  
DIBSK\_LENGTH = 00000074  
DIBSW\_UNIT = 0000000C  
DIBBUF = 000000D6 R 03  
DR750 = 000003CD R 02  
DR780 = 000003C9 R 02  
DRIOSTAT = 00000227 R 03  
DT\$ DR750 \*\*\*\*\* X 05  
DUMMY\_FAB = 0000173C R 03  
DUMMY\_RAB = 0000178C R 03  
DVIS\_DEVNAM = 00000020  
EFN1 = 00000001  
EFN2 = 00000004  
END\_UPDATE = 00000DE5 R 05  
ERROR\_COUNT = 0000014A R 03  
ERROR\_EXIT = 00000C70 R 05  
ERROR\_EXIT1 = 00000CCD R 05  
ERR\_FLGM = 00000008  
ERR\_FLGV = 00000003  
ESC = 0000001B  
EXIT\_DESC = 00000182 R 03  
EXIT\_HANDLER = 00000CF0 R 05  
FABSB\_BID = 00000000  
FABSB\_FNS = 00000034  
FABSC\_BID = 00000003  
FABSC\_BLN = 00000050  
FABSC\_SEQ = 00000000  
FABSC\_VAR = 00000002  
FABSL\_ALQ = 00000010  
FABSL\_DEV = 00000040  
FABSL\_FNA = 0000002C  
FABSL\_FOP = 00000004  
FABSL\_STS = 00000008  
FABSL\_STV = 0000000C  
FABSV\_CHAN\_MODE = 00000002  
FABSV\_CR = 00000001  
FABSV\_FILE\_MODE = 00000004  
FABSV\_GET = 00000001  
FABSV\_LNM\_MODE = 00000000  
FABSV\_PUT = 00000000  
FABSV\_UFO = 00000011  
FABSV\_UPD = 00000003  
FABSV\_UPI = 00000006  
FABSW\_GBC = 00000048  
FAO\_BUF = 0000000C R 03  
FILE = 0000041E R 02  
FIND\_IT = 000001E1 R 05

UE  
VO  
20  
64  
41  
66  
20  
20  
50  
70  
6F  
79  
6F  
53  
61  
66  
75  
6E  
61  
64  
20  
20  
75  
6F  
6F  
4C  
50  
74  
61  
65



UETDR7800  
Symbol table

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 54  
5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (19)

FLAG	0000000A	R	03
FOR\$CNV_OUT_F	*****	X	05
FOUND_IT	00000279	R	05
FPAC_FLGM	= 00000010		
FPAC_FLGV	= 00000004		
FREE	00000675	R	02
FREEQH	000012E8	R	03
FREE_PKT	00001598	R	03
GOBIT	000012CF	R	03
HALT	0000066E	R	02
HALT_PKT	00001310	R	03
HUNG_TEST	000008E1	R	05
ILLEGAL_REC	00000362	R	02
INADDRESS	0000015A	R	03
INDEV_UPDERR	000003D1	R	02
INI_FAB	00001650	R	03
INI_RAB	000016A0	R	03
INPTQH	000012D8	R	03
INPUT1_BUF	0000122F	R	03
INPUT_BUF	00000A2F	R	03
INPUT_ITMLST	00000195	R	02
IOSM_CTRLCAST	*****	X	05
IOS_READVBLK	*****	X	05
IOS_SETMODE	*****	X	05
IOS_STARTDATA	*****	X	05
IOCSGW_XFMXRATE	*****	X	05
IO_COMPLETE	0000092A	R	05
ITERATION	00000176	R	03
LC_BITM	= 00000020		
LIB\$SIGNAL	*****	X	05
MAX_DEV_DESIG	= 0000000A		
MAX_PROG_NAME	= 0000000F		
MAX_UNIT_DESIG	= 00000005		
MBCHAN	00000002	R	03
MODE	00000133	R	02
MSG_BLOCK	0000017E	R	03
NAME_LEN	= 0000000F		
NAME_TBL	000004E4	R	02
NEW_NODE	000001A0	R	03
NOOP	00000607	R	02
NOOP_PKT	000012F0	R	03
NOUNIT_SELECTED	0000033C	R	02
NO_CTRLNAME	000002D5	R	02
NO_OF_POS_PKTS	= 00000011		
NO_RMS_AST_TABLE	00000170	R	02
NRAT_LENGTH	= 00000014		
ONEMIN	000003FE	R	02
OTSS\$CVT_TI_L	*****	X	05
OUTADDRESS	00000162	R	03
OUTPUT_BUF	0000022F	R	03
PACK_REMOVED	000001AC	R	03
PAGES	= 00000009		
PASS	0000017A	R	03
PASS_MSG	00000396	R	02
PC1	00000A4C	R	05
PC1...	= 0000056C	R	02
PC2...	= 0000067C	R	02

PID	00000006	R	03
PKT1_AST	00000878	R	05
PKT_CHECK	00000A46	R	05
PKT_CNT	000001A8	R	03
PKT_COUNT	= 0000000D		
PKT_TBL	0000056C	R	02
PMT\$IZ	= 00000019		
PROCESS	00000041	R	02
PROCESS_NAME	000000A8	R	03
PROCESS_NAME FREE	= 0000000B		
PROC_CONT_NAME	0000008B	R	05
PROMPT	00000459	R	02
QUAD STATUS	00000152	R	03
RAB\$B_PSZ	= 00000034		
RAB\$B_RAC	= 0000001E		
RAB\$C_BID	= 00000001		
RAB\$C_BLN	= 00000044		
RAB\$C_BFA	= 00000002		
RAB\$C_SEQ	= 00000000		
RAB\$C_CTX	= 00000018		
RAB\$C_FAB	= 0000003C		
RAB\$C_PBF	= 00000030		
RAB\$C_ROP	= 00000004		
RAB\$C_STS	= 00000008		
RAB\$C_STV	= 0000000C		
RAB\$V_PMT	= 0000001E		
RAB\$W_RFA	= 00000010		
RAB\$W_RSZ	= 00000022		
RANBUF	00000A12	R	05
RANDOM1	0000016E	R	03
RANDOM2	00000172	R	03
RATE_BUF	00000208	R	03
RATE_DESC	0000021F	R	03
RATE_FLOAT	00000217	R	03
READ	000005A0	R	02
READ_CHAIN	000005A7	R	02
READ_CHA_PKT	000014A0	R	03
READ_DDI	0000062E	R	02
READ_DDI_PKT	000013B0	R	03
READ_PKT	000014D0	R	03
READ_SIZE	= 00000800		
RECORD	0000042A	R	02
REC_SIZE	= 00000028		
RESERVED	000005DC	R	02
RESTART	00000667	R	05
RMSS_BLN	*****	X	02
RMSS_BUSY	*****	X	02
RMSS_CDA	*****	X	02
RMSS_EOF	*****	X	05
RMSS_FAB	*****	X	02
RMSS_FACILITY	= 00000001		
RMSS_FNF	*****	X	05
RMSS_RAB	*****	X	02
RMS_ERROR	00000BCA	R	05
RMS_ERR_STRING	00000438	R	02
SAFE_TO_UPDM	= 00000004		
SAFE_TO_UPDV	= 00000002		

UE  
V0

67

72

20

62

20

60

66

45

45

49

49

5F

20

21

55

55

65

6C

20

20

20

21

65

20

73

69

72

55

59

55



UETDR7800  
Symbol table

M 1

VAX/VMS UETP DEVICE TEST FOR DR780/DR750 16-SEP-1984 00:21:03 VAX/VMS Macro V04-00 Page 55  
5-SEP-1984 04:35:16 [UETPSY.SRC]UETDR7800.MAR;1 (19)

SECSM_EXPREG	*****	X	05
SECSM_GBL	*****	X	05
SET_RAND_ENABLE	0000064A	R	02
SET_SELF_PKT	00001330	R	03
SET_SELF_TEST	000005E7	R	02
SHRS_ABEND	= 000010E0		
SHRS_BEGIN	= 00001038		
SHRS_ENDED	= 00001080		
SHRS_OPENIN	= 00001098		
SHRS_TEXT	= 00001130		
SS\$BADPARAM	= 00000014		
SS\$CONTROL	= 00000651		
SS\$NORMAL	= 00000001		
SS\$NOSUCHSEC	= 00000978		
SS\$SSFAIL	= 0000045C		
SS\$TIMEOUT	= 0000022C		
SS\$WASSET	= 00000009		
SSERROR	00000AE7	R	05
SS_SYNCH_EFN	= 00000003		
START_DATA_FAILED	0000028F	R	02
STATUS	0000014E	R	03
STR\$UPCASE	*****	X	05
STSSK_ERROR	= 00000002		
STSSK_INFO	= 00000003		
STSSK_SUCCESS	= 00000001		
STSSK_WARNING	= 00000000		
STSSM_INHIB_MSG	= 10000000		
STSS\$FAC_NO	= 0000000C		
STSS\$SEVERITY	= 00000003		
STSSV_FAC_NO	= 00000010		
STSSV_SEVERITY	= 00000000		
SUC_EXIT	0000098A	R	05
SUPDEV_GBLSEC	00000020	R	02
SUP_FAB	000016EC	R	03
SYSS\$ASSIGN	*****	GX	05
SYSS\$CANTIM	*****	GX	05
SYSS\$CLOSE	*****	GX	05
SYSS\$CONNECT	*****	GX	05
SYSS\$CREMBX	*****	GX	05
SYSS\$CREPRC	*****	GX	05
SYSS\$CRMPSC	*****	GX	05
SYSS\$DCLEXH	*****	GX	05
SYSS\$ERASE	*****	GX	05
SYSS\$EXIT	*****	GX	05
SYSS\$EXPREG	*****	GX	05
SYSS\$FAO	*****	X	05
SYSS\$FAOL	*****	GX	05
SYSS\$GET	*****	GX	05
SYSS\$GETCHN	*****	GX	05
SYSS\$GETDEV	*****	GX	05
SYSS\$GETDVI	*****	GX	05
SYSS\$GETMSG	*****	GX	05
SYSS\$INPUT	00000184	R	02
SYSS\$LKWSET	*****	GX	05
SYSS\$MGBLSC	*****	GX	05
SYSS\$OPEN	*****	GX	05
SYSS\$PUTMSG	*****	GX	05

SYSS\$QIO	*****	GX	05
SYSS\$QIOW	*****	GX	05
SYSS\$SETAST	*****	GX	05
SYSS\$SETEF	*****	GX	05
SYSS\$SETIMR	*****	GX	05
SYSS\$SETPRN	*****	GX	05
SYSS\$SETSFM	*****	GX	05
SYSS\$TRNLOG	*****	GX	05
SYSS\$UPDATE	*****	GX	05
SYSS\$WAITFR	*****	GX	05
SYSS\$WAKE	*****	GX	05
SYSIN_FAB	000015BC	R	03
SYSIN_RAB	0000160C	R	03
TENSEC	00000406	R	02
TERMQH	000012E0	R	03
TEST_DATA	0000022F	R	03
TEST_END	00000982	R	05
TEST_HUNG	000004A0	R	02
TEST_NAME	0000000F	R	02
TEST_OVERM	= 00000002		
TEST_OVERV	= 00000001		
TEXT_BUFFER	= 00000084		
THREEMIN	000003F6	R	02
TIME_IT	00000654	R	05
TTCHAN	00000000	R	03
UETDR7800	00000000	RG	05
UETP	= 00740000		
UETPS_ABEND	= 007410E0		
UETPS_ABORTC	= 0074832B		
UETPS_BEGIN	= 00741038		
UETPS_COPY_LOG	= 007480B1		
UETPS_COPY_LOG_ENDED	= 007480C1		
UETPS_COPY_LOG_LINE	= 007480B9		
UETPS_DATAER	= 00748010		
UETPS_DENOSU	= 00748333		
UETPS_ENDED	= 00741080		
UETPS_ERBOXPROC	= 00748020		
UETPS_FACILITY	= 00000074		
UETPS_OPENIN	= 00741098		
UETPS_TEXT	= 00741130		
UETUNT\$B_FLAGS	= 0000000B		
UETUNT\$B_TYPE	= 00000008		
UETUNT\$C_FAB	= 00000110		
UETUNT\$C_INDSIZ	= 000001A4		
UETUNT\$K_FAB	= 00000110		
UETUNT\$K_RAB	= 00000160		
UETUNT\$M_TESTABLE	= 00000002		
UETUNT\$T_FILSPC	= 00000014		
UETUNT\$V_TESTABLE	= 00000001		
UETUNT\$W_SIZE	= 00000009		
ULOAD_FAILED	00000271	R	02
UNIT_DESC	0000040E	R	02
UNIT_LIST	00000198	R	03
UNIT_LOOP	00000D62	R	05
UNIT_NUMBER	0000016A	R	03
UNUSED_FUNC	= 00006030		
UPDATE_FAILED	00000DC7	R	05

UE  
VO



WRITE	000005B4	R	02
WRITE_CHAIN	000005BC	R	02
WRITE_CH_PKT	000013D0	R	03
WRITE_DEV_CNTRL	000005CA	R	02
WRITE_PKT	00001400	R	03
WRITE_SIZE	= 00000800		
X	= 00000080		
XFSB_CMT_RATE	= 00000018		
XFSB_PKT_CMDCTL	= 0000000A		
XFSK_CMT_LENGTH	= 00000020		
XFSK_PKT_CBDMBC	= 00000003		
XFSK_PKT_CLRTST	= 00000007		
XFSK_PKT_DIAGRD	= 0000000B		
XFSK_PKT_DIAGRI	= 00000009		
XFSK_PKT_DIAGWC	= 0000000C		
XFSK_PKT_DIAGWI	= 0000000A		
XFSK_PKT_HALT	= 0000000F		
XFSK_PKT_NOINT	= 00000002		
XFSK_PKT_NOP	= 00000008		
XFSK_PKT_RD	= 00000000		
XFSK_PKT_RDCHN	= 00000001		
XFSK_PKT_SETTST	= 00000006		
XFSK_PKT_UNCOND	= 00000000		
XFSK_PKT_WRT	= 00000002		
XFSK_PKT_WRTCHN	= 00000003		
XFSL_PKT_DSL	= 0000001C		
XFSM_CMT_DIPEAB	= 00000002		
XFSM_CMT_SETRTE	= 00000001		
XFSM_PKT_FREQPK	= 00000008		
XFSV_PKT_FUNC	= 00000004		
XFSV_PKT_CISEL	= 00000003		
XFSV_PKT_FUNC	= 00000000		
XFSV_PKT_INTCTL	= 00000006		
XFLDR_COPY_FINISH	00000123	R	02
XFLDR_COPY_LINE	0000010F	R	02
XFLDR_COPY_START	000000F3	R	02
XFLDR_HUNG	00000077	R	02
XFLDR_LOG	000000AF	R	02
XFLDR_SYSSERROR	00000060	R	02
XFLDR_SYSSERROR_DESC	0000006F	R	02
XFLDR_SYSSERROR_FAB	000017D0	R	03
XFLDR_SYSSERROR_LENGTH	= 0000000F		
XFLDR_SYSSERROR_RAB	00001820	R	03

-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000067C ( 1660.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00001864 ( 6244.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
\$RMSNAM	00000023 ( 35.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DR78	00000E1B ( 3611.)	05 ( 5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE



+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	37	00:00:00.07	00:00:00.87
Command processing	138	00:00:00.73	00:00:03.63
Pass 1	606	00:00:27.66	00:00:56.53
Symbol table sort	0	00:00:02.43	00:00:04.94
Pass 2	551	00:00:07.45	00:00:12.63
Symbol table output	46	00:00:00.34	00:00:00.44
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1382	00:00:38.72	00:01:19.08

The working set limit was 2000 pages.  
165248 bytes (323 pages) of virtual memory were used to buffer the intermediate code.  
There were 90 pages of symbol table space allocated to hold 1637 non-local and 79 local symbols.  
2154 source lines were read in Pass 1, producing 45 object records in Pass 2.  
71 pages of virtual memory were used to define 63 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SHRLIB]UETP.MLB;1	2
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	56
TOTALS (all libraries)	58

1959 GETS were required to define 58 macros.  
There were no errors, warnings or information messages.  
MACRO/LIS=LIS\$:UETDR7800/OBJ=OBJ\$:UETDR7800 MSRC\$:UETDR7800/UPDATE=(ENH\$:UETDR7800)+EXECML\$/LIB+SHRLIB\$:UETP/LIB



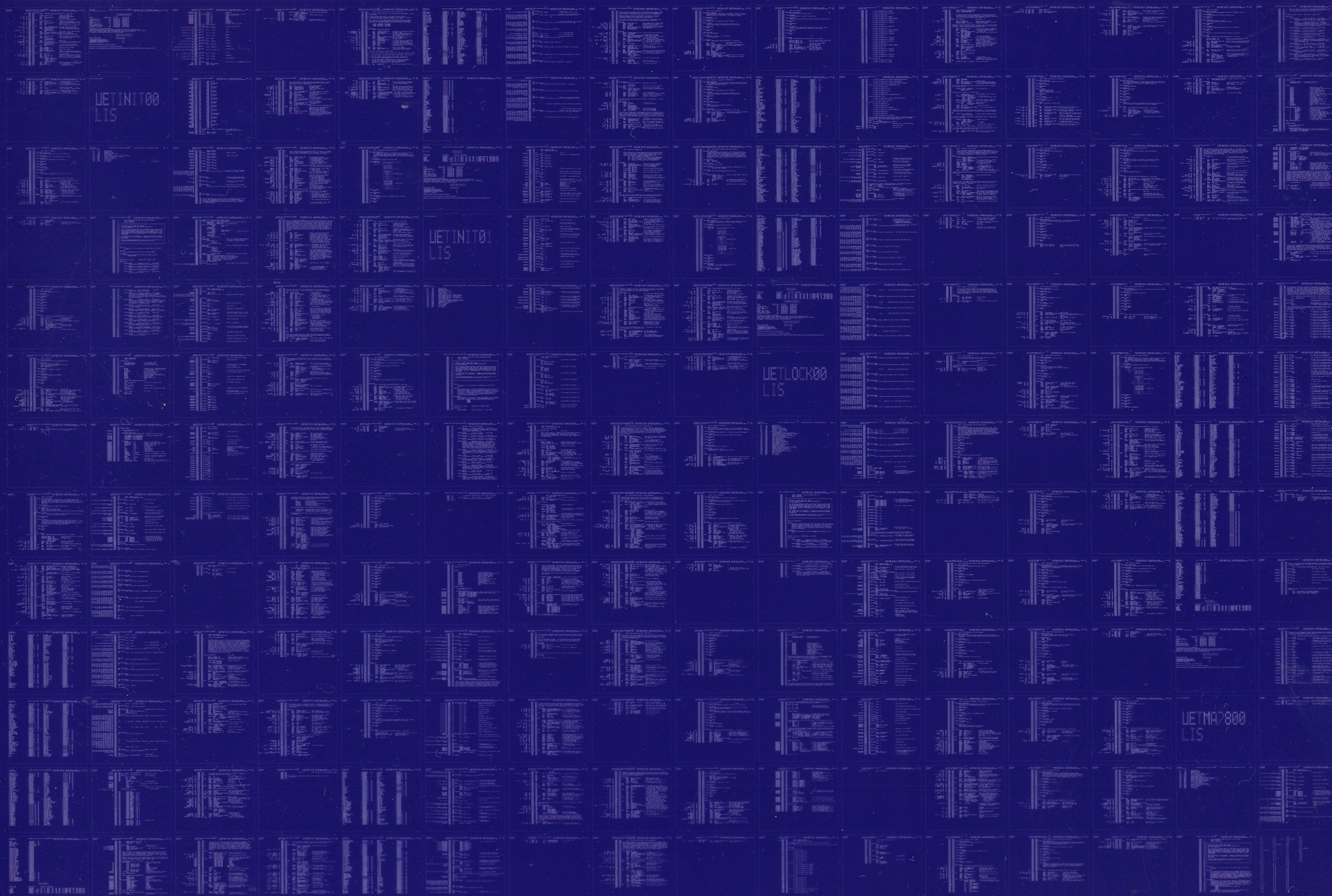
0426 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0427 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



UETMA7800  
LIS